KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Remigijus Kiminas

Ekstremaliu programavimu pagrįstos projektų valdymo sistemos prototipas ir planavimo etapo empirinis tyrimas

Magistro darbas

Darbo vadovas
dr. Tomas Blažauskas

Kaunas, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Remigijus Kiminas

Ekstremaliu programavimu pagrįstos projektų valdymo sistemos prototipas ir planavimo etapo empirinis tyrimas

Magistro darbas

Vadovas
dr. T. Blažauskas
2006-05

Atliko
IFM-3/2 gr. stud.
Remigijus Kiminas
2009-05-25

Recenzentė
doc. dr. Lina Čeponienė
2009-05

Kaunas, 2009

Turinys

Santrauka

Pastaruoju metu ypač populiarėja lanksčiojo programavimo šeimos programinės įrangos kūrimo procesų modeliai. Šiame darbe aprašoma realizuota PĮ, kuri remiasi ekstremalaus programavimo metodika. Sukurta PĮ leidžia lengviau kontroliuoti PĮ kūrimo proceso eigą. Realizuoti pagrindiniai ekstremalaus programavimo metodikoje aprašyti PĮ kūrimo etapai**.** Atliktas empirinis tyrimas, siekiant nustatyti planavimo etapo charakteristikas konkrečioje įmonėje. Planavimo etapas daro didžiausią įtaką projekto biudžetui ir sėkmei. Neteisingas projekto trukmės įvertinimas ne tik uždelsia projekto įgyvendinimą, bet ir atneša nuostolius klientui ir projektą vykdančiai įmonei. Buvo nustatyta, kad vidutinis darbo trukmės prognozavimo tikslumas yra apie 70%. Empirinio tyrimo metu nustatyta, kad didžiausią įtaką planuojamo darbo trukmei ir užgaišto laiko skirtumui turi ne programuotojų netikslūs trukmės įvertinimai, bet priežiūros darbai, kurie trijų savaičių stebėjimo laikotarpyje sudarė apie 30% kalendorinio laiko, todėl buvo įtakojami projektų pabaigos terminai.

Remigijus K. Extreme programming based project management system prototype and an empirical study of the planning phase: Master's work in science of informatics supervisor dr. T. Blažauskas; Department of Software Engineering, Faculty of Informatics, Kaunas University of Technology. – Kaunas, 2009.

Summary

This work describes the realized project management system, which is based on extreme programming methodology. Developed software makes it easier to control the software creation process. Extreme programming features have been implemented in the developed software. Was conducted an empirical study to determine the characteristics of the planning phase of a particular company. The planning phase has the greatest impact on project budget, success. An incorrect estimate of duration of the project has an impact on project delivery dates and lead to additional costs for the company, which is responsible for this. It was found that the accuracy of project estimates is about 70%. With empirical studies we have found that the support has the greatest impact on the software project estimates. Using developed software the ideal time was monitored. Within three weeks of the experiment, it was found that the support is about 30% of the calendar work, which has a significant impact on the time estimations of the project.

Paveikslėlių sąrašas

Lentelių sąrašas

1. ĮVADAS

Efektyvus projektų valdymas reikalauja sisteminio proceso ir jį palaikančių įrankių, kurie struktūrizuotų šį procesą ir leistų koordinuoti projekto eigą. Projektų valdymas, ypač programinės įrangos kūrimo, sudėtingas procesas, apimantis vykdytojus, užsakovus ir darbų paskirstymas tarp vykdytojų. Projektų valdymo programinė įranga turi užtikrinti darnų komandos darbą su minimaliomis laiko sąnaudomis. Šiose sistemose turi būti realizuoti pagrindiniai projektų vykdymo etapai, kaip užduočių paskirstymas, rizikos valdymas. Šios sistemos leidžia kompiuterizuoti projektų valdymo procesus ir metodologijas, užtikrinti projektų savalaikiškumą [1]. Pastaruoju metu ypač populiarėja lanksčiojo programavimo programų kūrimo metodologijos. Populiariausia šio tipo metodologija – ekstremalus programavimas [31].

Kuriant tokias sistemas, būtina atsižvelgti ir į rizikos valdymą. Nuolatinis išorės IT specialistų („Outsourcing“) vykdomų projektų skaičius auga, todėl būtina atkreipti dėmesį ir į potencialią riziką, kuri atsiranda šio proceso metu [2].

Tyrimo objektas – ekstremalaus programavimo metodikos planavimo etapas.

Pagrindinis darbo tikslas – tai ekstremalaus programavimo („Extreme programming“) metodika pagrįstos projektų valdymo sistemos prototipo sukūrimas ir planavimo etapo empirinis tyrinimas, pasinaudojant sukurtu prototipu.

Šiame darbe keliami šie uždaviniai:

- Egzistuojančių programinės įrangos kūrimo procesų analizė ir ją palaikančios programinės įrangos analizė.
- Pasirinkto programinės įrangos kūrimo proceso analizė, esminių savybių išskyrimas. Detalus planavimo esybių išskyrimas.
- Projektų valdymo sistemos prototipo sukūrimas, kuris remtųsi pasirinktu programinės įrangos kūrimo proceso modeliu.
- Sistemos diegimo tyrimas įmonėje ir planavimo etapo empirinis tyrimas.

Primoje darbo dalyje nagrinėjami egzistuojantys sprendimai. Apibendrinamos vyraujančios tendencijos projektų valdyme. Išskiriami programinės įrangos kūrimo procesų modeliai, klasifikacija. Pagrindžiamas tokių sistemų būtinumas. Apžvelgiamos sukurtos sistemos, išskiriamos pagrindinės jų savybės ir klasifikacija. Pagrindžiamos projektų valdymo sistemos kūrimui naudojamos technologijos.

Projektinėje dalyje akcentuojami priimti esminiai architektūriniai sprendimai kuriant sistemą. Detalizuojamos pagrindinės sistemos komponentės ir jų tarpusavio sąveika. Atskleidžiama sąsaja tarp ekstremalaus programavimo metodikos ir realizuotos projektų

valdymo sistemos. Akcentuojamos esminės sukurtos sistemos savybės ir jų panaudojimo galimybės ekstremalaus programavimo planavimo etapo eigoje.

Programinės įrangos diegimo tyrime apžvelgiamos sukurtos sistemos savybės. Analizuojami vartotojų pateikti siūlymai, jų pagrįstumas. Tiriamoji dalis skirta įvertinti programos patogumui naudojantis ja darbo aplinkoje ir skirtingoms vartotojų grupėms.

Eksperimentinė dalis skirta empiriškai įvertinti kaip tiksliai dabartinėje situacijoje įvertinamos darbų trukmės ir faktoriai įtakojantys juos. Kelių savaičių laikotarpyje įmonėje buvo fiksuojami atliekami darbai sukurtos sistemos pagalba. Po atliktų stebėjimų buvo išanalizuoti gauti rezultatai ir įvertintos planavimo etapo metrikos – projekto greitis, vartotojo pasakojimų fiksavimas, idealus laikas, užimtumo koeficientas. Įvertinus gautus duomenis pateikti siūlymai planavimo etapo charakteristikų gerinimui, patogesniam planavimo etapo stebėjimui.

2. PROJEKTŲ VALDYMO SISTEMŲ ANALIZĖ

2.1. Problema

Remiantis „Standish“ grupės duomenimis 1994 metais tik 16 % projektų buvo sėkmingi, o 2000 – 28 %, 2002 – 34 %, 2004 – 29 %, 2009 – 32 % [3,7,42]. Iš šių skaičių matyti, kad projektų sėkmės koeficientas sąlyginai mažas. Projektas laikomas sėkmingu, kai tenkinami šie reikalavimai:

- Projektas baigtas laiku.
- Neviršytas biudžetas.
- Realizuotos visos funkcijos įvardintos projekto dokumentacijoje.

Tos pačios grupės duomenimis 60% sėkmingų projektų atlikusių bendrovių naudojo projektų valdymo sistemas [3]. Tokios programinės įrangos naudojimas tampa vienu iš esminių įrankių, užtikrinančių projekto sėkmę.

Šiuolaikiniams projektams vis didesnę įtaką turi išorės IT paslaugų pirkimas. Darosi vis ryškesnė įtaka PĮ kūrimo etapams. 1998 metais atliktos apklausos duomenimis 70 % kompanijų, kurios naudojosi išorinių kompanijų resursais, buvo patenkintos atliktais darbais [8]. Tačiau atsiranda ir nemaža problemų, kaip kultūriniai skirtumai, laiko zonos skirtumai. Būtina patikrinti kompanijų ar darbuotojų patikimumą [5]. Atsiranda būtinybė užtikrinti gerą komunikacinį ryšį tarp projekto vykdytojų [4], o norint integruoti išorinius resursus į projektų valdymo programinę įrangą, būtina užtikrinti šios programinės įrangos saugumą [6].

Anglijoje atliktame programinės įrangos rizikos vertinimo empiriniame tyrime buvo išskirtos dažniausiai sutinkamos rizikos, [43] įtakojančios projekto baigtį. Dažniausia pasitaikančios rizikos pateiktos 1 lentelėje.

1 lentelė. Dažniausia pasitaikančios rizikos [43]

Rizika	**Dažnai**	**Vidutiniškai**	**Retai**
Neatitikimas laiko vertinimams ir biudžetui	X		
Nuolatinė reikalavimų kaita	X		
Neaiški arba neteisingai suprasta sritis/objektai	X		
Patyrusio projekto vadovo trūkumas	X		
Kliento neįtraukimas į PĮ kūrimo procesą		X	
Ne taip suprasti reikalavimai		X	
„Gold plating“		X	
Efektyvios projekto valdymo metodologijos trūkumas			X
Neadekvačios žinios/įgūdžiai			X
Kuriamos blogos programinės įrangos funkcijos			X

Buvo išskirtos ir svarbiausios rizikos empirinio tyrimo metu. Rezultatai pateikti 2 lentelėje.

2 lentelė. Didžiausią įtaką turinčios rizikos [43]

Rizika	**Bendras vertinimas**	**Patirtis 2-5 m.**	**Patirtis 6-10 m.**	**Patirtis >10 metų.**
Neaiški arba neteisingai suprasta sritis/objektai	1	1	1	1
Ne taip suprasti reikalavimai	2	2	2	8
Kliento neįtraukimas į PĮ kūrimo procesą	3	3	6	3
Patyrusio projekto vadovo trūkumas	4	4	5	4
Kuriamos blogos programinės įrangos funkcijos	5	5	3	10
Neatitikimas laiko vertinimams ir biudžetui	6	7	4	2
Nuolatinė reikalavimų kaita	7	6	9	7
Neadekvačios žinios/įgūdžiai	8	8	8	5
Efektyvios projekto valdymo metodologijos	9	9	7	6
„Gold plating“	10	10	10	9

Kaip matyti iš atlikto tyrimo, didžiausią įtaką projekto eigai turi *neaiški arba neteisingai suprasta sritis*. Dažniausiai pasikartojanti rizika buvo įvardinta – *neatitikimas laiko vertinimams ir biudžetui*. Rizikos mažinimui naudojamos įvairios projektų valdymo metodikos.

Lanksčiojo programavimo šeimos metodikos akcentuojamos į dinamišką projekto vykdymą. Pateiktuose duomenyse tai atitinka nuolatinę reikalavimų kaitą. Kadangi projektas vykdomas dinamiškai ir sąlyginai nedidelėmis iteracijos, žymiai sumažėja rizikos, susijusios su nuolatine reikalavimų kaita, neatitikimams laiko vertinimams, neteisingai suprastais reikalavimais. Į procesą įtraukiamas ir vartotojas – tai leidžia lengviau suprasti nagrinėjamą sritį, o vartotojo grįžtamasis ryšys užtikrina kuriamų funkcijų korektiškumą. Atlikus pirminę literatūros analizę rastos kelios projektų valdymo sistemos, kurios remtųsi lanksčiojo programavimo metodikomis. Tačiau jose buvo realizuotos minimalios funkcijos, susijusios su

planavimo etapu. Realizuotas funkcijas galima panaudoti tik konkrečioje integruotoje kūrimo aplinkoje. Kitose sistemose buvo akcentuojamos bendros proceso savybės. Kadangi ekstremaliojo programavimo metodikoje vienas iš pagrindinių etapų yra planavimo etapas [36], buvo sukurtas prototipas, leidžiantis realiu laiku fiksuoti planavimo etapo charakteristikas, kurių fiksavimas nepriklauso nuo to, su kokia programine įranga yra dirbama. Prototipo naudojimas turėtų žymiai palengvinti planavimo etapo duomenų kaupimą ir analizę. Sukurtame prototipe automatiškai apskaičiuojamos idealus laikas ir užimtumo koeficientas. Buvo realizuota ir išoriniams vartotojams skirta žiniatinklio sąsaja, kurios pagalba ateityje būtų galima įtraukti į PĮ kūrimo procesą ir išorinius resursus.

2.2. Programinės įrangos kūrimo procesų modeliai

Egzistuoja daugiau nei dešimt programinės įrangos kūrimo procesų modelių. Įvardinkime pagrindinius [15]:

- „Agile“ – lanksčiojo programavimo metodologijų šeima
- „Unicycle“
- „Code-and-Fix“
- „V-Methodology“
- „Waterfall“ – krioklio modelis
- „Open Source“
- „Spiral“ – spiralinis modelis
- „Synchronize and Stabilize“
- „Reverse Engineering Development“ – atvirkštinė inžinerija
- „General Publication“
- „Structured Systems Analysis and Design Method“
- „The Pramis“
- „Offshore Development“
- „General Drug Development“

Nagrinėdami literatūrą, galime išskirti dvi pagrindines metodologijų šakas:

- Iteracinis – kiekvienos iteracijos metu leidžiama nauja programos versija, tenkinanti minimalius kliento reikalavimus.
- Inkrementinis – metodologija remiasi palaipsniniu produkto realizavimu.

Microsoft kompanijos kuriami produktai remiasi „Synchronize and Stabilize“ metodologija [17]. Priklausomai nuo pasirinktos projektų valdymo metodologijos reikalingi ir atskiri įrankiai. Naudojant „Offshore Development“ metodologiją mums reikės įrankių, leidžiančių bendrauti tarpusavyje tarp nutolusių projekto vykdytojų. Įmanomas ir metodologijų apjungimas. Šiuo pagrindu buvo sukurta „*dX process*“ metodologija apjungianti XP ir RUP procesus [15, 19]. Kurią metodologiją pasirinkti, priklauso nuo daugybės faktorių, kaip projekto sudėtingumas, komandos sudėtis, laiko apribojimai ir kiti.

Pasauliniu mastu vis populiarėja lanksčiojo programavimo programinės įrangos kūrimo metodologijos ir jos atmainos. Viena populiariausių šio tipo metodologijų yra ekstremalaus programavimo metodologija [31]. Pagrindinė problema su kuria susiduriama integruojant išorinių paslaugų pirkimą ir ekstremalaus programavimo metodiką – bendravimo stoka [32]. Nors šia tema ir yra atliekami tyrimai, tačiau ši sritis nėra visapusiškai išnagrinėta [33].

2.3. PĮ projektų sėkmės faktoriai

Nagrinėjant projektų valdymo metodiką galime išskirti šiuos sėkmės faktorius [9, 12]:

1. Projekto palaikymas – ši sąvoka apima projekto užsakovą ir jo skiriamas lėšas projekto vystymui.
2. Būsimo vartotojo įtraukimas į projekto eigą – galima daug sužinoti ir įvertinti, jei į projekto eigą būna įtraukiamas užsakovas. Jis padeda greičiau perprasti veikimo srities ontologiją ir pašalinti loginius konfliktus.
3. Patyręs projektų vadovas.
4. Pagrindinių projekto tikslų nustatymas.
5. Projekto apimties nustatymas.
6. Projekto skaidymas į smulkesnes užduotis.
7. Aiškiai apibrėžtas projekto valdymo procesas.
8. Automatizuotų įrankių naudojimas.
9. Nuolatinis rizikos vertinimas.

Projektuojant ir kuriant šiuolaikinę projektų valdymo programinę įrangą būtina atsižvelgti į šiuos faktorius.

2.4. Projektų valdymo sistemų klasifikacija

Egzistuoja kelios pagrindinės projektų valdymo sistemų architektūros [10]:

- „Desktop“ – tai dažniausiai vartotojo kompiuteryje veikianti programa.

- „Web“ aplikacijos – programų sistemos, kurių nereikia diegti vartotojo kompiuteryje. Jos veikia interneto naršyklės pagalba.
- „Personal“ – tai dažniausiai namuose ar asmeniniams tikslams naudojama projektų valdymo sistema.
- „Single User“ – tai vienam vartotojui skirta projektų valdymo sistema, kur projekto planą ir kitus veiksmus gali atlikti tik vienas asmuo.
- „Collaborative“ – šios sistemos suteikia galimybę turėti keletą vartotojų, kurie atsakingi už tam tikrą projekto dalį. Realizuojamas bendravimas tarp projekto vykdytojų, užsakovų ir programuotojų.
- „Integrated“ – šios sistemos apima CRM, užduočių sąrašus, įvykių kalendorius.

2.5. Pasirinktas programinės įrangos kūrimo procesas

Apžvelgus programinės įrangos kūrimo metodikas buvo nuspręsta, kad kuriama programinė įranga bus grindžiama ekstremalaus programavimo („Extreme Programming“) metodika. Ši metodika literatūroje dar vadinama ribiniu programavimu. Šiame darbe mes naudosime ekstremaliojo programavimo terminą. Šioje metodikoje akcentuojamos trys pagrindinės dalys. Programinės įrangos kūrimo procesą sudaro projektas apimantis visas sudedamąsias projekto vykdomas dalis. Iteracijas, kurios grindžiamos vartotojo pasakojimais. Darbais, kurie remiasi iteracijose aprašytais vartotojo pasakojimais.

Ši metodika buvo pasirinkta, nes dažniausiai kuriant nedidelės apimties žiniatinklio sistemas nebūna rašomi detalūs reikalavimai. Pradiniuose etapuose įvardinamos esminės sistemos funkcijos, kurios, kūrimo etapuose būna detalizuojamos vartotojo pasakojimų pagalba. Tapatūs veiksmai atliekami, kai būna planuojama darbų trukmė jau įgyvendintiems projektams. Tai įtakojo ekstremalaus programavimo metodikos pasirinkimą, kaip pagrindą kuriamai sistemai.

2.6. Ekstremalaus programavimo proceso analizė

Siekiant sukurti projektų valdymo sistemą, kuri remtųsi ekstremaliu programavimu, būtina pastarojo proceso analizė ir esminių esybių išskyrimas. Ekstremalaus programavimo metodika buvo pristatyta 1999 m. Kent Beck [34]. Tai populiariausia lanksčiojo programavimo šeimos metodika [35]. Bazinės ekstremalaus programavimo praktikos išskaidomas į keturias grupes [32,44,45]:

- Pastovus grįžtamasis ryšys
    i. Testais paremtas programavimas

    ii. Planavimo žaidimas/etapas
    iii. Vientisa komanda įtraukiant užsakovą
    iv. Programavimas poromis
- Nenutrūkstamas procesas
    i. Nuolatinis integravimas
    ii. Pertvarkymas
    iii. Smulkios laidos
- Bendras supratimas
    i. Paprastas projektas
    ii. Sistemos metafora
    iii. Bendra kodo nuosavybė
    iv. Kodavimo standartai
- Programuotojų gerovė
    i. Pastovus tempas

Mes detaliau panagrinėsime planavimo etapo esminius aspektus, kadangi atliekamas darbas akcentuojamas į planavimo etapo empirinį tyrimą. Planavimo etapas dar yra vadinamas planavimo žaidimu [44,45]. Planavimo etapas yra vienas iš esminių ekstremalaus programavimo metodikos etapų [36]. Planavimo etapo metu nustatomos darbų trukmės, įvertinamos darbų trukmės remiantis anksčiau atliktais darbais. Beck. K. akcentuoja du laiko vertinimo gerinimo būdus tai:

- Laikytis paprastumo principo („Keep it simple“)
- Vertinti darbus anksčiau atliktais darbais („Yesterday‘s weather“)

Norint vertinti darbus anksčiau atliktų darbų atžvilgiu – būtina juos fiksuoti. Ekstremaliame programavime naudojamos vartotojo pasakojimų istorijų korteles [37]. Istorijos kortelės pavyzdys pateiktas 9.2 priedo punkte. Šiose istorijų kortelėse fiksuojami vartotojo pasakojimai („User story“), planuojamos ir realios darbų trukmės.

Darbų trukmės vertinimai nebūtinai gali remtis laiko sąnaudomis, bet gali būti naudojami kiti matavimai, kaip vartotojo pasakojimui priskirti taškai. Jei naudojamas kalendorinės pastangos [36], naudojamas idealus laikas („Ideal time“). Idealaus laiko sąvoką galime apibrėžti, kaip sugaištą laiką atliekant konkrečią užduotį be trukdymų.

Minima dar viena planavimo etapo sąvoka – projekto greitaveika („Project velocity“). Apibrėžime teigiama, kad projekto greitis yra realizuotų vartotojo pasakojimų skaičius iteracijos eigoje [40,41]. Santykis tarp kalendorinio laiko ir realaus sugaišto laiko/idealaus

laiko [38, 39] vadinamas užimtumo koeficientu. Tiek vienas, tiek kitas dydis gali būti empiriškai išmatuotas [39]. Kalendorinis laikas – tai aštuonios darbo valandos per dieną.

Užimtumo koeficientas apskaičiuojamas pasitelkus šią formulę:

Užimtumo koeficientas = Kalendorinių dienų skaičius / Idealių dienų skaičiaus

Apibendrindami galime išskirti šias pagrindines planavimo etapo charakteristikas

- Idealus laikas
- Projekto greitis
- Vartotojo pasakojimai
- Užimtumo koeficientas

Norint padidinti projekto greitaveiką, tiek padidinti prognozuojamų darbų trukmės vertinimo tikslumą, būtina kaupti aukščiau išvardintus parametrus. Šiuos parametrus nepatogu kaupti popierinėje formoje, kortelių pavidalu. Didelių problemų gali kilti, jei reikia fiksuoti idealų laiką dienos eigoje, kai atliekama nemažai papildomų darbų. Kuriant programinę įrangą bus siekiama automatizuoti vartotojo pasakojimų kaupimą. Tai leis atsisakyti nereikalingo kortelių pildymo. Jeigu darbų trukmės būtų fiksuojamos kortelėse, būtų sunku įvertinti idealaus laiko trukmes iteracijų atžvilgiu, kadangi į idealų darbą neįeina visi pašaliniai darbai atliekami dienos metu. Automatiškai surinktos informacijos pagalba bus galima apskaičiuoti išvardintus parametrus. Toliau apžvelgsime keletą projektų valdymo sistemų, kurios būtų akcentuotos į lankstų programinės įrangos kūrimo procesą.

2.7. Projektų valdymo prototipų apžvalga

Siekiant optimizuotu projektų valdymo procesą, atliekami vis nauji moksliniai tyrimai, kuriami prototipai pagrįsti programinės įrangos kūrimo procesais.

2.7.1. Massachusetts‘o Kembridžo atliktas tyrimas ir sukurtas prototipas

Šiame tyrime analizuojamos dabartinių projektų valdymų sistemų ypatybės ir esminiai jų trūkumai. Daroma išvada, kad dauguma projektų valdymo sistemų nesprendžia vartotojų bendradarbiavimo problemos. Norint pasiekti vieną ar kitą tikslą, dažnai pasitelkiamos kelios programos: el. pašto, tiesioginio bendravimo ir kitos. Atsiranda sklaida tarp projektų valdymo įrankių. Buvo bandoma apjungti formalią ir neformalią projektų valdymo metodologiją, praplečiant projekto užduoties sąvoką – veiklos sąvoka. Kiekviena užduotis galėjo būti išskaidoma į keletą veiklos gijų, sudaromos jų hierarchijos.

Atliktame tyrime dalyvavo 17 projektų vadovų, kurių vidutinė patirtis projektų valdyme yra 6 metai. Buvo stebimas jų darbas, analizuojami naudojami įrankiai, problemos su kuriomis buvo susiduriama. Po trijų, mėnesių remiantis tyrimų rezultatais, buvo sukurtas prototipas „Activity Centered Project Management“ ACPM [11].

Programinė įranga nuo įprastinių projektų valdymo sistemų skyrėsi šiais aspektais:

1. Vartotojai galėjo tarpusavyje bendrauti pokalbių kambaryje.
2. Įdiegta galimybė dalintis failais tarpusavyje.
3. Įdiegta automatiška vartotojų informavimo sistema RSS pagalba. Kadangi iš atliktų tyrimų paaiškėjo, jog 44 % laiko projektų vadovai praleidžia sekdami projektų būseną ir 19 % informuodami klientą apie projekto eigą. Buvo pasiūlytą keletas skirtingų RSS naujienų šaltinių, priklausomai nuo vartotojo reikalavimų.
4. Buvo sprendžiamos išorės darbuotojų problemos, pasitelkus žiniatinklio technologijas, kur vartotojo įvesta informacija buvo sinchronizuojama su duomenų serveriu. Projekto vadovas galėjo tiesiai iš projektų valdymo sistemos išsiųsti el. pašto žinutę su užklausimu. Žinutės turinyje buvo automatiškai formuojama nuoroda, kurią paspaudęs vartotojas galėjo atsakyti į projekto vadovo užklausimą. Gauta informacija buvo sinchronizuojama su projektų valdymo sistema.

   Pristatytas prototipas susilaukė nemažo susidomėjimo, tačiau jis dar tobulinamas.

2.7.2. DrProject – edukacinis projektų valdymo portalas

Tai Toronto universiteto sukurtas projektų valdymo portalas edukaciniais tikslais. Pagrindinis projekto tikslas – skatinti darbą komandoje. Šiame projektų valdymo portale realizuotos šios galimybės [13]:

- Wiki – jis susiejo į visumą visas naudojamas portalo dedamąsias
- Versijų valdymas
- Užduotys („Tickets“)
- Elektroninių laiškų siuntimo eilės
- Projekto eigos žingsnių nustatymas

Projekto pabaigoje padarytos šios išvados:

- Šis projektas leido lavinti komandinio darbo įgūdžius, analizuoti projekto eigą
- Palengvino dėstytojų darbą

Sukurtas prototipas labiausiai atitinka integruotų projektų valdymo architektūra.

2.7.3. XPSuite

Šis įrankis integruoja ekstremaliu programavimu grįstą projektų valdymą ir jo būsenos sekimą. Jis sudarytas iš dviejų pagrindinių komponentų:

- XPSwiki – šis įrankis skirtas valdyti XP projektų valdymą. Jis buvo realizuotas pasitelkus žiniatinklio technologijas.
- XP4IDE – įskiepis skirtas integruoti XPSwiki į IDE IntelliJ-Ide.

Pagrindinis tyrimo tikslas -surinkti XP metodika paremto projekto eigos metriką. Pasinaudodami įskiepiu jie galėjo stebėti šias projekto charakteristikas:

- Artefakto aktyvacijos pokytį (suminis laikas praleistas artefakto langui esant aktyviam.)
- Praleistą laika programuojant tam tikrą artefaktą.
- Užduoties aktyvumo laikas (bendras praleistas laikas programuojant šią užduotį) susumuojant bendrą užduoties artefaktų langų aktyvumo laiką.)
- Suminį praleistą laiką programuojant visus užduoties artefaktus.
- Bendrą laiką susumavus užduoties aktyvacijos ir programavimo laikus.

Šiuo atveju artefaktus galime vadinti darbus, susijusius su užduotimis. Šie darbai turi būti realizuoti, norint įvykdyti pasirinktą užduotį.

Atlikus tyrimą padarytos tokios išvados [14]:

- Naudojant tokią metodologiją, reikia atkreipti dėmesį į programuotojų laisvę, kadangi ši metodologija gali būti naudojama sekti darbuotojų veiksmams, o ne projekto efektyvumo didinimui.

Naudojant šią metodologija įmanoma aiškiai sekti su užduotimis ir artefaktais išeikvotą laiką. Galima aiškiai matyti laiko sąnaudas, susijusias su užduotimis. Tai ypač svarbu, kadangi naudojant ekstremalaus programavimo metodologiją projektas įgyvendinamas per daug mažų iteracijų, kur klientas apmoka už sugaišta laiką kiekvienos iteracijos pabaigoje [15].

2.8. Realizavimo technologijos

Apžvelkime galimas projektų valdymo sistemų architektūrinius aspektus.

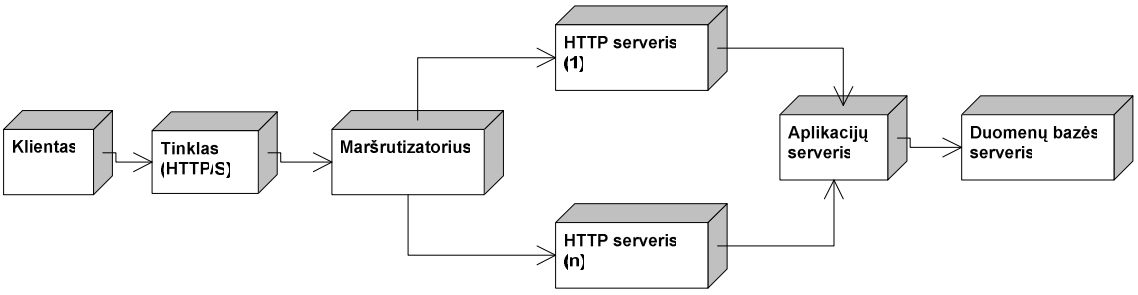
2.8.1. Žiniatinklio sprendimas

Šis sprendimas gali būti realizuojamas pasitelkiant žiniatinklio kūrime naudojamas technologijas. Dažniausiai – tai interneto naršyklės lange veikiančios programos. Šios programos pasirenkamos dėl šių faktorių [21]:

1. Pakankamai lengvai realizuojamos ir diegiamos.
2. Nesudėtingas programinės įrangos atnaujinimas. Nebrangus sistemų palaikymas.
3. Nereikalingi papildomi diegimai vartotojo kompiuteryje. Tik vartotojo kompiuteris turi būti prijungtas prie interneto.
4. Centralizuotas duomenų saugojimas.

Šios aplikacijos realizuojamos pasitelkus trijų lygių architektūrą [20]:

1. Interneto naršyklė
2. Žiniatinklio serveris
3. Duomenų bazės serveris



1 pav. Žiniatinklio aplikacijų architektūra

Šios programos turi ir trūkumų [22]:

1. Naudojami HTML elementai gali atrodyti skirtingai priklausomai nuo naudojamos interneto naršyklės.
2. Sunku naudoti įvedamų duomenų kaukes, kaip telefono numerio formatas, el. pašto adresas. Visais atvejais būtinas papildomas programavimas „Javascript“.
3. Ribotos HTML elementų pozicijos nustatymo galimybės.
4. Veikimo sparta gali būti mažesnė negu įprastinių ekrano programų, nes kiekvienos užklausos metu persiunčiamas puslapis iš naujo, o ne nauji duomenys.
5. Integravimas su kitais produktais iš esmės labai komplikuotas ir bet kuriuo atveju jau būna kuriama įprastinė programa, veikianti vartotojo kompiuteryje.

2.8.2. Vartotojo kompiuteryje veikiančios programos

Šio tipo programinė įranga būna diegiama vartotojų kompiuteriuose. Kaip pagrindinius tokios programinės įrangos privalumus galime išskirti [22]:

1. Platesnės galimybės kuriant programinės įrangos vartotojo sąsają.
2. Suteikiama galimybė išnaudoti visus kompiuterio resursus – vaizdo plokštes, procesorius ir kitus.
3. Veikimo sparta yra didesnė negu žiniatinklio aplikacijų, nes programos langas sugeneruojamas tik vieną kartą ir nereikia jo pergeneruoti kiekvienos užklausos metu.
4. Integracija su kitais produktais tampa lengviau realizuojama.

Tačiau ir jos turi trūkumų [22]:

1. Reikalingas programinės įrangos diegimas vartotojo kompiuteryje.
2. Programų palaikymas brangiai kainuoja, kadangi atnaujinimai turi būti parsisiunčiami kiekvieno vartotojo atskirai.

2.8.3. Saugumo užtikrinimas

Nesvarbu, kurią realizavimo metodiką pasirinkus, būtina užtikrinti naudojamos programinės įrangos saugumą. Visų pirma turi būti užtikrintas ryšio saugumas tarp kliento ir serverio. Tam tikslui pasitelkiami SSL, TLS protokolai [23, 24]. Būtina užtikrinti ir pačios programinės įrangos saugumą, PL/SQL procedūrų naudojimas padidina [25, 26, 27]:

1. Sistemos bendrą integralumą.
2. Programos įrangos saugumą.
3. Programos efektyvumą.

Sistema, kurios dalis funkcijų realizuota DBVS lygyje, gali būti lengvai keičiama vartotojo sąsaja. Sumažėja klaidų skaičius, kadangi mes vieną sykį ištestuojame funkciją ir ją galime naudoti keliose programos vietose. Nemažesnis faktorius tampa ir tinklo apkrovimas, šiuo atvejų, kai naudojamos PL/SQL procedūros, vartotojo programa siunčia vieną užklausą, o PL/SQL funkcijoje gali būti realizuotos kelios SQL užklausos [28].

2.9. Pasirinkta realizavimo architektūra

Kuriamą sistemą sudaro dvi dalys:

- Vartotojų kompiuteriuose veikianti programinė įranga

- Žiniatinklio aplikacija

Vartotojų kompiuteriuose veikianti programinė įranga bus atsakinga už realiu laiku vykstantį laiko fiksavimą – idealaus laiko fiksavimą. Joje bus kaupiami visi svarbiausi projekto parametrai. Surinktų parametrų pagalba bus galima apskaičiuoti pagrindines planavimo etapo metrikas.

Išanalizavus realizavimo technologijos pasirinktas serverio ir kliento architektūros principas. Šiuo atveju duomenų bazės serveris bus „PostgreSQL“ DBVS. Klientinę dalis bus kuriama „QT“ C++ aplinkoje. Šis pasirinkimas suteiks galimybę programų sistemos vartotojui dirbti nepriklausomai nuo platformos. Klientai tiesiogiai jungsis prie DBVS. „PostgreSQL“ vartotojų sistema suteikia plačias galimybes realizuojant tokios architektūros programas, nes [29]:

1. Turi integruotą procedūrinę SQL programavimo kalbą, vadinama PL/pgSQL.
2. Suteikiama galimybė patiems apibūdinti duomenų tipus.
3. Galima praplėsti sistemą savo apibūdintom funkcijom, naudojant C ar C++ programavimo kalbas.
4. Galima aprašyti vaizdus („views“).

Tokia architektūra leis lengvai praplėsti sistemą naujomis funkcijomis ir moduliais.

Kaip „QT“ karkaso privalumus galima išskirti [30]:

1. Nepriklausoma nuo platformos.
2. Turi tvarkyklių rinkinį, leidžiantį jungtis prie MySQL, PostgreSQL ir kitų DBVS.
3. Turi stabilią API sąsają.

Žiniatinklio aplikacija bus realizuota pasinaudojus „Zend Framework“ programinės įrangos paketu. Šis programavimo karkasas turi visas būtinas klases greitam PĮ kūrimui. Žiniatinklio aplikacija bus atsakinga už išorinių resursų informavimą apie projekto eigą RSS pagalba. Jos pagalba bus galima išreikšti komentarus apie atliekamus darbus. Pagrindinis žiniatinklio aplikacijos tikslas – grįžtamasis ryšys iš kliento pusės. Žiniatinklio aplikacijos teikiamas RSS srautas gali pasitarnauti ir kaip dokumentacijos šaltinis klientui, kadangi programuotojai jo pagalba gali aprašyti, kaip veikia konkrečios sistemos funkcinės dalys. Nebūtina atskirai gaišti laiko projekto vadovui aprašant sistemos funkcines savybes, kadangi jos bus aprašytos programuotojų, kuriant PĮ.

2.10. Literatūros analizės išvados

Atliktos analizės rezultatus galime apibendrinti priimtais sprendimais:

- Realizuota projektų valdymo sistema remsis ekstremalaus programavimo metodika, kuri yra lanksčiojo programavimo šeimos vienas iš metodų.
- Sukurto prototipo pagalba turi būti galima apskaičiuoti planavimo etapo pagrindinės charakteristikas.
- Bus atliktas ekstremalaus programavimo planavimo etapo empirinis tyrimas, įvertintos planavimo etapo charakteristikos ir jų panaudojimas planuojant darbų trukmes. Empirinis tyrimas bus atliktas „Coral solutions“ įmonėje.
- Apžvelgus sukurtus prototipus buvo pastebėta, kad mažas dėmesys skiriamas planavimo etapo charakteristikų fiksavimui. Realizuotas XP4IDE įrankis suteikia minimalias galimybes planavimo etapo analizei. Negalima analizuoti rezultatų projekto, iteracijų atžvilgiu. Įrankio naudojimas galimas tik IDE IntelliJ-Ide aplinkoje.
- Sukurtą programinę įrangą sudarys dvi dalys
    i. Vartotojų kompiuteryje veikianti PĮ
    ii. Žiniatinklio sąsajoje veikianti programinė įranga

Tolimesniuose skyriuose bus detalizuoti priimti architektūriniai sprendimai bei pateikti empirinio tyrimo rezultatai.

3. PROJEKTŲ VALDYMO SISTEMOS PROTOTIPAS

Šiame skyriuje aprašyti esminiai sukurtos sistemos architektūriniai sprendimai ir detalizuota projektų valdymo sistemos sąsaja su ekstremalaus programavimo metodika.

3.1. Ekstremalaus programavimo esybių analizė

Prieš pradedant kurti sistemą atliktas detalus ekstremalaus programavimo modelio esybių išskyrimas ir išskirtos šios pagrindinės esybės [34]:

- Projektas
- Iteracija
- Darbas

Siekiant padaryti kuriamą programinę įrangą lankstesnę įvedami papildomi atributai projekto esybei. Projektas papildytas šiais atributais:

- Aprašymu – aprašymas skirtas pateikti bendro pobūdžio informacijai apie projektą.
- Būsenomis – būsenos naudojamos indikuoti projekto eigą.

- RSS šaltiniai – RSS šaltinių pagalba galima greitai sužinoti projekto eigą arba realizuotų atskirų projekto funkcinių dalių veikimą. RSS šaltiniai gali būti naudojami dokumentacijos pateikimui.
- Sekcija – sekcijų pagrindu ribojamos vartotojų teisės.
- Klientas – kiekvienam projektui priskiriamas klientas.
- Biudžetas – biudžeto pagalba apskaičiuojamas vidutinis projekto valandinis įkainis.

Realizuotas projekto langas pateiktas 9.3 priedo punkte.

Sekantis žingsnis buvo išskirti iteracijos atributus. Atributų išskyrimui panaudota informacija pateikta [34] šaltinyje, pasakojimo kortelių šablono pagalba 9.2 priedo dalis. Išskirti šie atributai:

- Data
- Istorijos numeris
- Veikos rūšis
- Prioritetas
- Vartotojas
- Rizika
- Laiko įvertinimas
- Darbo aprašymas
- Pastabos
- Darbai

Pateikti atributai yra bendro pobūdžio.ir todėl atributų sąrašą buvo nuspręsta papildyti šiais atributais:

- Galimybe planuoti iteracijos trukmę pasinaudojant planuojamų darbų trukmėmis. Pasinaudojant šia savybe galima skaidyti vartotojo pasakojimą į smulkesnius darbus ir tuo pačiu metu automatiškai skaičiuoti iteracijos planuojamą trukmę.
- Pabaigos data – darbo pabaigos data naudojama generuojant ataskaitas.
- Vartotojo pasakojimų versijų numeravimu – versijų numeravimas suteikia galimybę, nekurti naujos iteracijos, o publikuoti vartotojo pasakojimą kaip atskirą versiją. Kiekvienas darbas yra siejamas su konkrečia iteracijos pasakojimo versija.
- Prisegamais failais – kadangi kompiuterinė programa leidžia dirbti ne tik su tekstine informacija, buvo sukurta galimybė prisegti failus prie iteracijos.

- Automatiškai formuojama statistika, kuri leidžia pažiūrėti suminį darbų idealų laiką.

Realizuoto langas pateiktas 9.4 priedo punkte.

Sekančiame etape išskirti darbo atributai. Remiantis 9.2 priede pateikta informacija buvo išskirti šie darbo atributai:

- Data
- Būsena
- Numatomi darbai
- Komentaras

Siekiant papildyti pirminius darbo atributus buvo įvestos naujos savybės:

- Atsakingas asmuo
- Komentarai
- Sukūrimo data
- Pabaigos data
- Planuojama trukmė
- Idealus laikas

Darbo lango realizacija pateiktas 9.5 priede.

Išskyrus esamas ir naujas savybes galima pereiti prie konkrečių reikalavimų formavimo kuriamai sistemai. Tolimesniame skyriuje detalizuosime sistemai keliamus reikalavimus, atsižvelgiant į išskirtas savybes. Bus detalizuota pasirinkta sistemos architektūra ir detalizuoti sukurto prototipo architektūriniai sprendimai.

3.2. Reikalavimai sistemai

3.2.1. Sistemos tikslai

Sistema turėtų leisti susisteminti žiniatinklio projektus vykdančios įmonės veiklą, dalinai atsižvelgiant į šios srities specifines veiklos charakteristikas. Galima išskirti šiuos pagrindinius sistemos tikslus:

1. Laiko sąnaudų sumažinimas efektyviai paskirstant darbus tarp programuotojų. Tai galima pasiekti savaitės eigoje stebint kiekvieno programuotojo užimtumo koeficientą.
2. Žiniatinklio puslapių tobulinimo sąnaudų analizė remiantis anksčiau atliktais darbais, kliento pasakojimu.
3. Esminių ekstremalaus programavimo metodikos savybių pritaikytas žiniatinklio projektų vykdyme.

4. Projekto analizė trukmės atžvilgiu.
5. Sistemos architektūroje turi būti numatyta galimybė teikti projektų valdymo paslaugas kitoms įmonėms.
6. Pasakojimo kortelės transformacija į programinės įrangos sąsają.

3.2.2. Vartotojai

Numatytos keturios vartotojų grupės:

Projektų vadovas – tai projektą koordinuojantys asmenys, atsakingi už projekto eigą. Jiems aktualus lengvas projekto koordinavimas ir užgaišto laiko analizė.

Programuotojas – jiems aktualus gaišto laiko fiksavimas ir kliento pasakojimų saugojimas. Reikalavimų kaupimas kuriamai sistemai. Dokumentacijos pateikimas projekto vadovams.

Dizaineris – tai grafinius darbus atliekantys asmenys. Jiems aktualus lengvas laiko fiksavimas. Reikalavimų kaupimas.

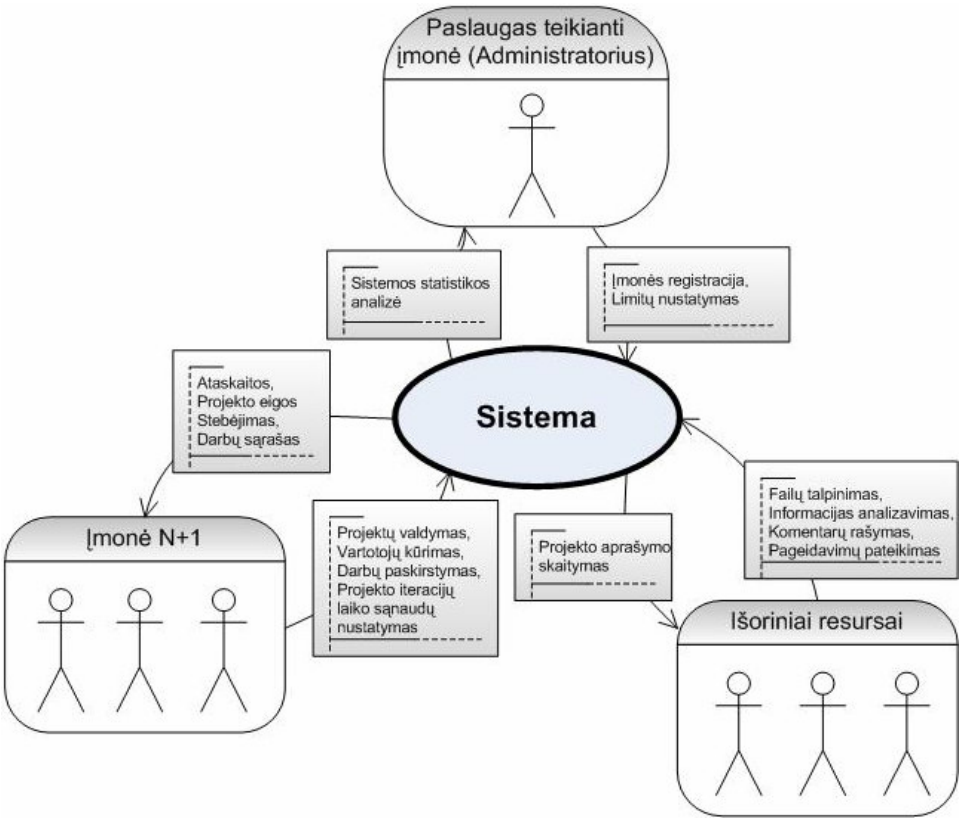
Skyriaus vadovas/direktorius – aktualios ataskaitos susijusios su pinigų srautais.

3.2.3. Diegimo aplinka

PĮ bus diegiama žiniatinklio projektus vykdančios įmonės aplinkoje. Daugumoje darbo vietų įdiegta Windows XP ir Windows Vista operacinės sistemos. Programinė įranga turi veikti tiek stacionariame, tiek nešiojamame kompiuteryje.

3.2.4. Veiklos kontekstas

Pateiktame veiklos konteksto paveikslėlyje (2 pav.), veiklos konteksto diagramoje pateikiama, kaip sistema sąveikauja su ją naudojančiais vartotojais.



2 pav. Veiklos kontekstas

3 lentelė. Veiklos įvykių sąrašas

Eil. Nr.	***Įvykio pavadinimas***	***Įeinantys/Išeinantys informacijos srautai***
1	Administratorius registruoja naują įmonę	Duomenys apie įmonę *(in)* Įmonei nustatomi limitai *(in)*
2	Įmonės administratorius registruoja naują vartotoją	Suveda vartotojo parametrus *(in)*
3	Įmonės darbuotojas peržiūri skirtų darbų sąrašą	Darbų sąrašo peržiūra *(out)*
4	Projektų vadovas registruoja naują projektą	Projekto parametrų nustatymas *(in)*
5	Išorinis resursas dalyvauja projekto vykdymo eigoje	Siunčia failą *(in)* Stebi projekto eigą (RSS) *(out)* Pateikia pageidavimus *(in*)
6	Projektų vadovas valdo projekto vykdymo procesą	Generuojamos ataskaitos *(out)* Paskirsto darbus *(in)* Aprašo projekto duomenis *(in)*

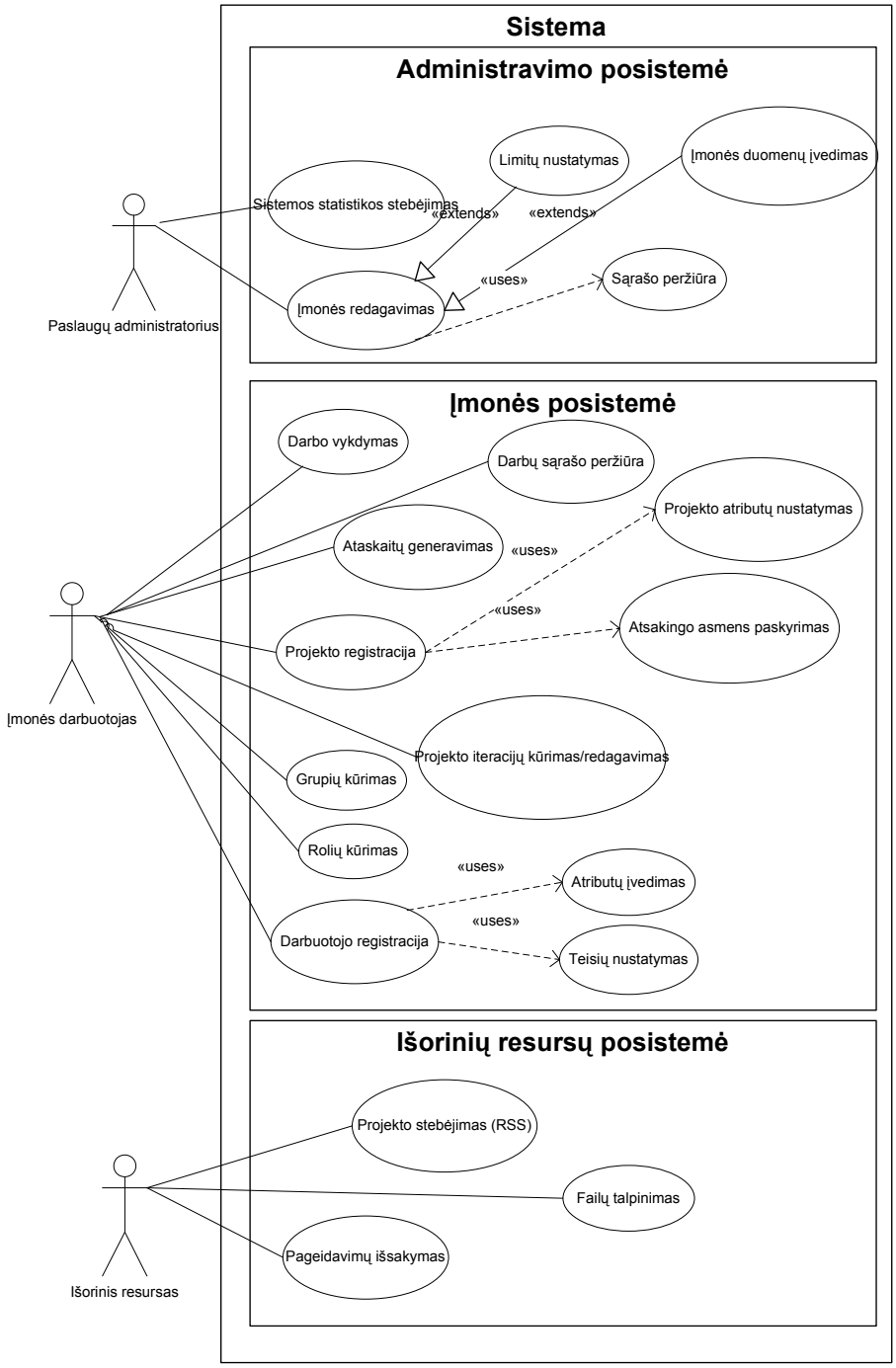
Veiklos kontekstas atskleidžia, kaip realizuotas sistemos veikimo principas. Jis sudarytas iš trijų esminių grupių:

- Įmonių, kurios naudosis šia sistema

- Išorinių resursų
- Administratorių, kurie galės užregistruoti naują įmonę, kuri naudosis šia sistema.

Galimybė registruoti naujas įmonės buvo įvesta tam, kad ateityje sukurtą programinę įrangą būtų galima nuomoti kitoms įmonėms. Kuriant prototipą ši savybė buvo panaudota projektuojant programinės įrangos architektūrą ir realizacijoje prototipo lygmenyje.

3.2.5. Panaudojimo atvejai



3 pav. Panaudojimo atvejų diagrama

Panaudojimo atvejus sudaro trys pagrindinės panaudojimo atvejų grupės:

1. Administratoriaus panaudojimo atvejai
2. Įmonės posistemės panaudojimo atvejai
3. Išorinių resursų posistemės panaudojimo atvejai

Administratoriaus panaudojimo atvejai

- Įmonės limitų nustatymas
- Įmonės duomenų įvedimas
- Sistemos statistikos stebėjimas
- Įmonės redagavimas
- Įmonių sąrašo peržiūra

Įmonės darbuotojų panaudojimo atvejai

- Idealaus laiko fiksavimas realiu laiku
- Projekto atributų nustatymas
- Projekto atsakingo asmens priskyrimas
- Projekto iteracijų aprašymas
- Darbų sąrašo peržiūra
- Projekto skaidymas į iteracijas, naujos iteracijos kūrimas remiantis vartotojo pasakojimu
- Ataskaitų generavimas
- Darbuotojo registracija
- Vartotojo teisių nustatymas
- Darbuotojo atributų nustatymas
- Failų talpinimas
- Vartotojų grupės kūrimas, vaidmenų priskyrimas grupėms
- Vartotojų vaidmenų kūrimas

Išorinių resursų panaudojimo atvejai

- Projekto eigos stebėjimas RSS
- Pageidavimų išsakymas

- Failų talpinimas

3.2.6. Funkciniai reikalavimai

Funkcinius reikalavimus sudaro trys pagrindinės grupės:

1. Administratoriaus reikalavimai
2. Išorinių vartotojų reikalavimai
3. PĮ naudojančios įmonės reikalavimai

Administratoriaus reikalavimai:

- Sistema turi leisti nurodyti, kiek projektų gali turėti įmonė
- Turi būti galima nustatyti, kiek vartotojų gali turėti įmonė
- Turi būti galima peržiūrėti įmonių sąrašą
- Turi būti sudaryta galimybė redaguoti pasirinktą įmonę

Įmonės administratorius keliami reikalavimai:

- Sistema turi leisti susigeneruoti norimą ataskaitą
- Sistema turi leisti priskirti vartotojams vaidmenis
- Administratorius gali sukurtą neribotą vaidmenų skaičių
- Administratorius gali kurti grupes
- Administratorius gali priskirti vaidmenis grupėms
- Sistema turi leisti nustatyti, kuriai grupei priklauso vartotojas

Įmonės darbuotojų keliami reikalavimai

- Turi būti galima realiu laiku fiksuoti užgaištą laiką
- Vartotojas gali sukurti projektą, jei jam suteiktos atitinkamos teisės
- Vartotojas turi galėtu peržiūrėti jam priskirtus darbus
- Vartotojas turi galėti išrikiuoti darbus pagal norimą darbo atributą
- Turi būti numatyta galimybė filtruoti projektus pagal klientą
- Sistema turi leisti išskaidyti projektą į norimas iteracijas
- Turi būti galima suformuoti ataskaitą, kurioje būtų įvertintas planuojamas ir realus užgaištas laikas

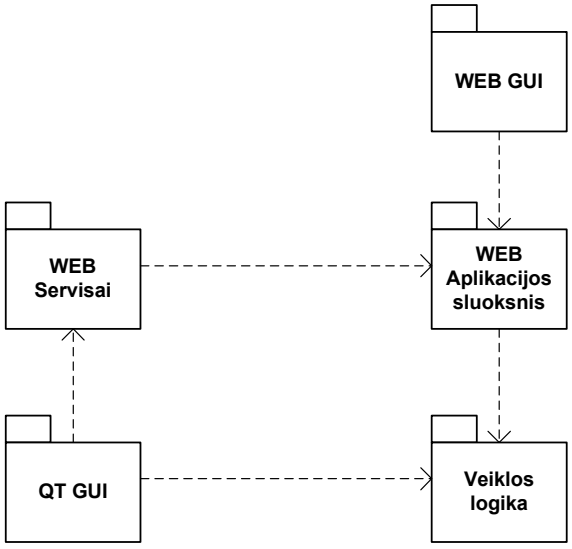
3.2.7. Nefunkciniai reikalavimai

Nefunkciniais reikalavimais siekiama apibrėžti reikalavimus sistemos išvaizdai, panaudojamumui, vykdymo charakteristikoms, veikimo sąlygoms ir pan. Prieiga prie projektų turi būti galima iš dviejų sąsajų :

- Žiniatinklio
- Vartotojų kompiuteriuose veikiančios PĮ
- Laiko sąnaudos saugomos minučių tikslumu.
- Finansinius duomenis būtina saugoti dviejų skaičių po kablelio tikslumu
- Teisių sistema turi būti realizuojama DBVS lygyje. Tokiu būdu bus taupomas interneto srautas ir išvengta tų pačių funkcijų dubliavimo žiniatinklio ir vartotojų kompiuteryje veikiančioje PĮ.
- Sistema neturi saugoti su vartotojais susijusių duomenų, kurių saugojimui reikalingi specialūs leidimai.

3.3. Sistemos architektūra

3.3.1. Sistemos statinis vaizdas



4 pav. Bendra paketų diagrama

Sistemą sudaro du vartotojo sąsajos paketai:

- Vartotojų kompiuteryje veikianti programinė įranga
- Žiniatinklio sąsaja

Vartotojų kompiuteryje veikianti programinė įranga bus atsakinga už ekstremalaus programavimo metodikos realizaciją vartotojų lygmenyje. Joje bus realizuotos šios dalys:

- Projektas

- Iteracija
- Darbas/užduotis

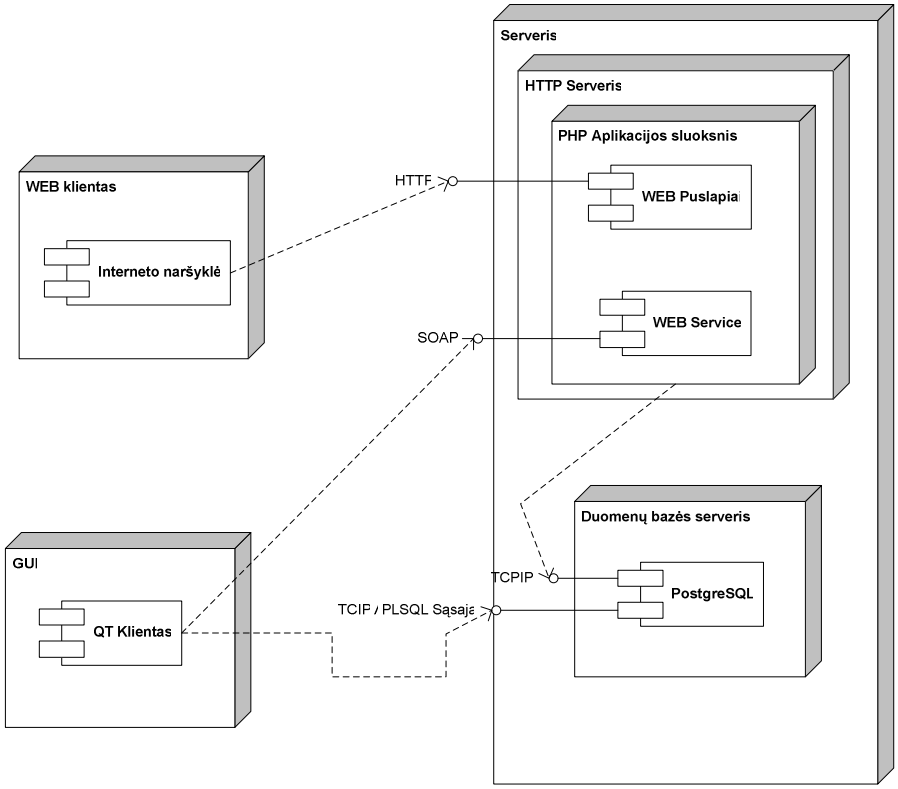
Vidinė architektūra sudaro trys paketai:

- Veiklos logika realizuota reliacinėje duomenų bazėje
- Žiniatinklio serviso tarnyba skirta failų siuntimui į serverį
- Žiniatinklio aplikacijos branduolys

Vidinė architektūra bus atsakinga už šias esmines funkcija

- Informacijos saugojimą
- Failų talpinimą
- Teisių sistemos realizaciją

3.3.2. Išdėstymo vaizdas



5 pav. Išdėstymo diagrama

Sistemą sudaro trys pagrindiniai sistemos komponentai:

- QT Klientas – tai vartotojų kompiuteriuose veikianti programinė įranga
- Reliacinė duomenų bazė

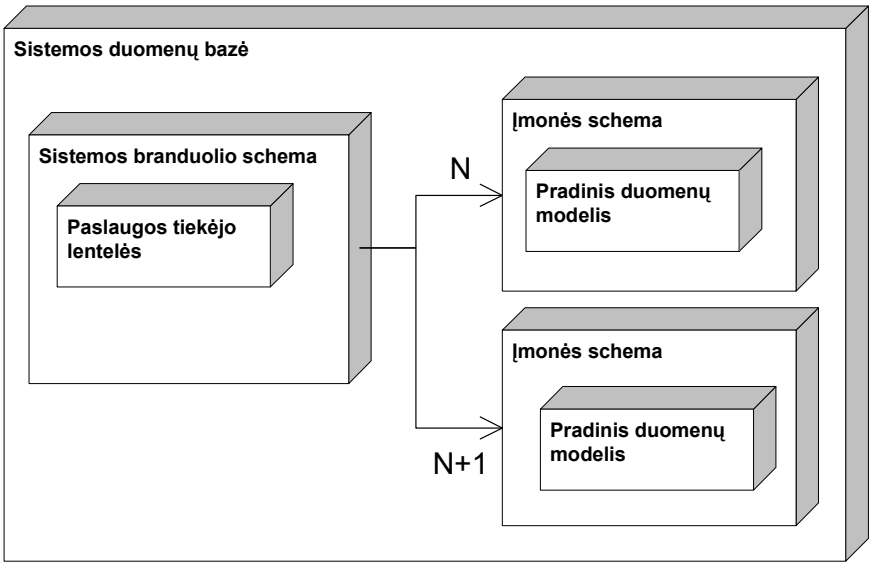
- Interneto naršyklė
- Žiniatinklio aplikacijos sluoksnis

3.3.3. Duomenų bazės modelis

Konteksto modelyje buvo apibrėžta galimybė teikti paslaugas kelioms įmonėms vienu metu. Ši savybė buvo realizuojama pasinaudojus PostgreSQL teikiamoms schemų savybėmis. Sistemos duomenų bazė padalinta į dvi schemas:

- Paslaugos tiekėjo duomenų lenteles
- Įmonių lentelės. Kiekviena įmonė gauna savo „Schema“ ir lenteles joje.

Tokia abstrakcija pateikta 6 pav.

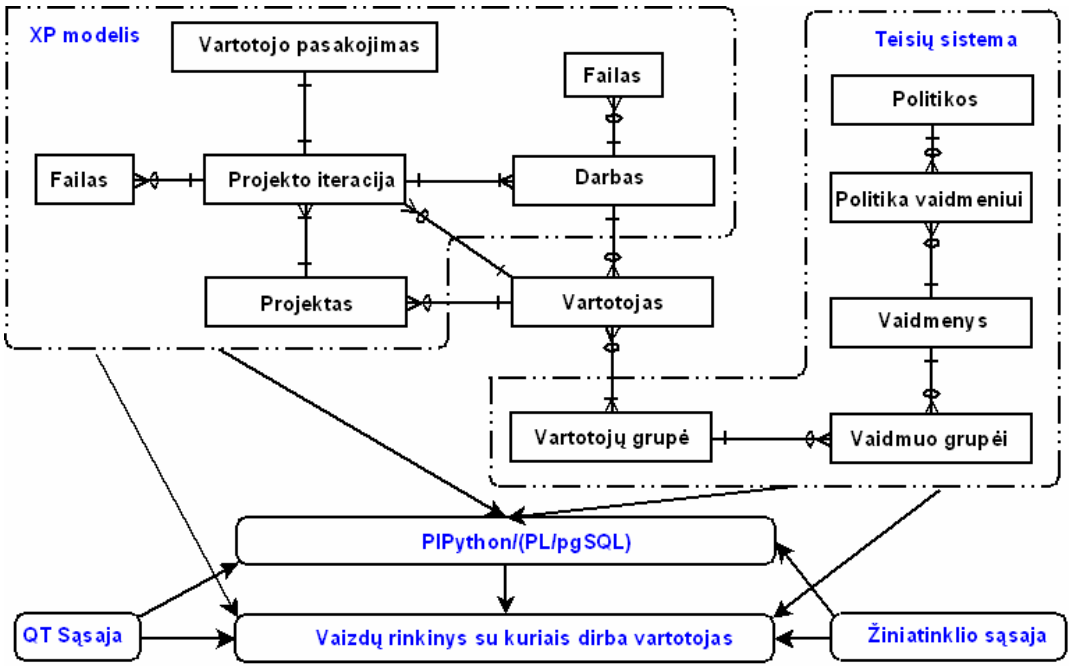


6 pav. Duomenų bazės struktūra schemų forma

Paslaugos tiekėjo lentelėse buvo realizuota limitų sistema. Limitų sistemos pagalba nurodoma, kiek įmonė gali turėtu projektų, vartotojų, iteracijų. Šioje dalyje saugoma informacija apie užregistruotas įmones. Talpinama informacija apie sukurtus modulius ir jose realizuotas funkcijas. Šioje schemoje buvo realizuotos vartotojų teisių tikrinimo funkcijos Python programavimo kalba. Toks sistemos padalinimas į vieną pagrindinę schema ir įmonių schemas leidžia:

1. Centralizuotai vykdyti sistemos atnaujinimo darbus
2. Diegti naujus modulius
3. Sukurti paskyrą naujai įmonei, neatliekant techninių darbų
4. Pasiūlyti įvairesnių paslaugų įmonėms

Pradiniame duomenų modelyje realizuota ekstremalaus programavimo koncepcija. Bendrinė schema pateikta 7 pav.



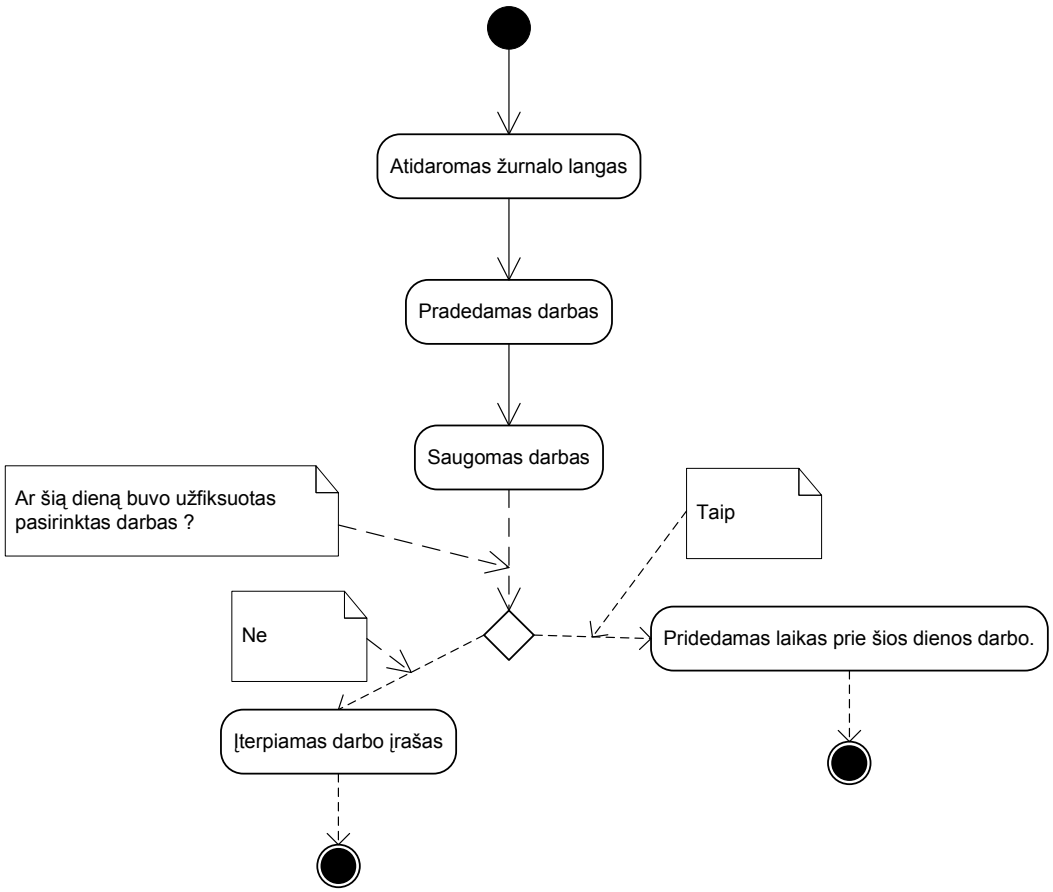
7 pav. Pradinis duomenų modelis

Pasirinktas duomenų modelis leido lanksčiai realizuoti teisių sistemą. Sudarytą duomenų modelį sudaro du pagrindiniai komponentai:

- Ekstremalaus programavimo modelio lentelės
- Teisių sistemos lentelės

Paveikslėlyje pateikta, kaip sąveikauja duomenų bazėje realizuotos sistemos funkcijos su programinės įrangos komponentais. Šitoks duomenų modelis leidžia lanksčiai praplėsti sistemos funkcionalumą atliekant minimalius pataisymas kituose programinės įrangos sluoksniuose.

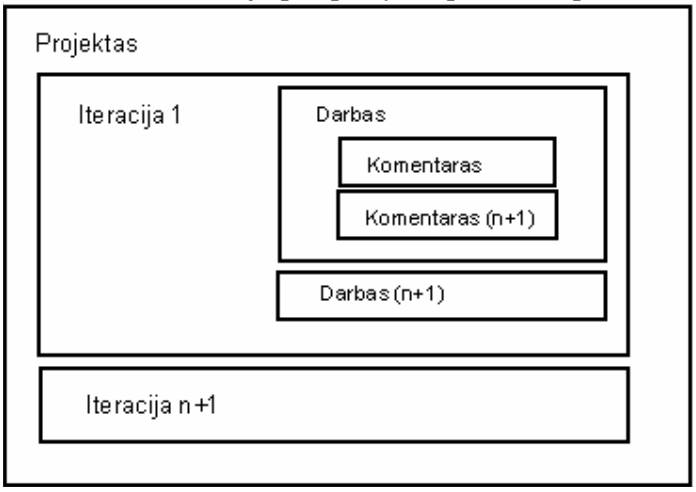
Ekstremalaus programavimo dalyje realizuotos idealaus darbo sekimo funkcijos. Šis funkcionalumas realizuojamas pasinaudojus trigeriais. Idealaus darbo fiksavimo trigerio veikimas pateiktas žemiau esančiame paveikslėlyje.



8 pav. Žurnalo funkcijos realizacija

Pateiktas funkcijos realizavimo algoritmas leidžia išvengti darbų įrašų dubliavimo, kai pasirinktas darbas būna stabdomas, pradedamas kelis kartus per dieną.

Sistemos architektūra iš vartotojo perspektyvos pateikta 9 paveikslėlyje.



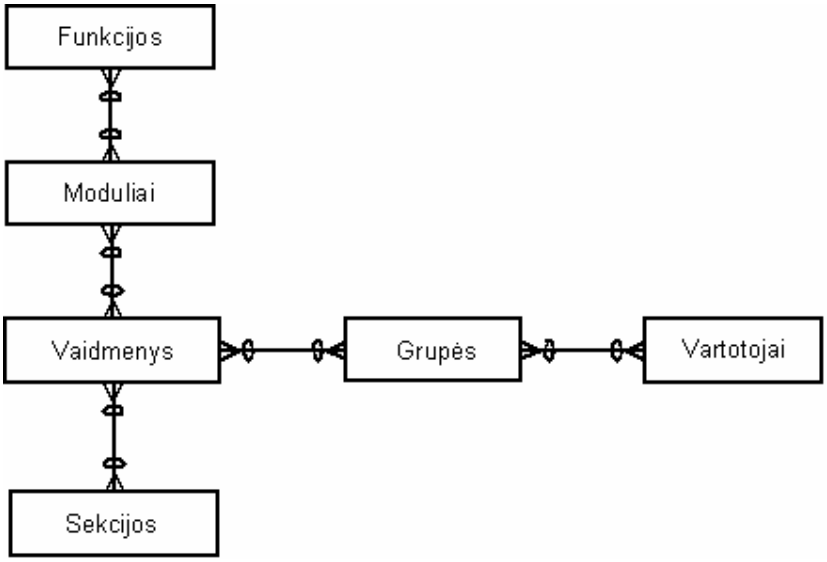
9 pav. Sistemos koncepcinis modelis

3.3.4. Teisių sistemos modelis

Vartotojo teisių sistemą sudaro penkti komponentai:

- Grupės
- Vartotojai
- Vaidmenys
- Moduliai
- Funkcijos
- Sekcijos

Prieš detaliau paaiškindami teisių sistemos veikimą, pateiksime esminį teisių sistemos modelį.



10 pav. Teisių sistemos modelis

Modulius sudaro tam tikri funkcijų rinkiniai. Funkcijų pagalba mes galima suteikti teisę skaityti darbus, bet neleisti redaguoti. Sukurtame prototipe realizuojami šie pagrindiniai moduliai

- Projektas
- Iteracijas
- Darbas
- Komentaras

Kiekvienas modulis su priskirtomis funkcijomis gali būti priskirtas vaidmeniui. Tokiu būdu mes galime suformuoti skirtingus vaidmenis su skirtingomis teisėmis. Vaidmenį gali

sudaryti skirtingi moduliai ir funkcijos. Vaidmenų teises galime nesunkiai koreguoti priskirdami naujai suskurtus modulius su skirtingomis funkcijomis.

Vaidmenys gali būti ribojami sekcijų atžvilgiu. Sekciją galime suprasti, kaip apribojimą, taikomą kuriam nors vaidmeniui. Kiekvienas vaidmuo gali būti ribojamas kelių sekcijų atžvilgiu. Atskiras sekcijas gali turėti šie objektai:

- Projektas
- Iteracija
- Darbas

Tokiu būdu kiekvienas vartotojas gali leisti atskiras projekto dalis, pvz.: dizaino darbus, redaguoti tik dizaineriui ar panašiai.

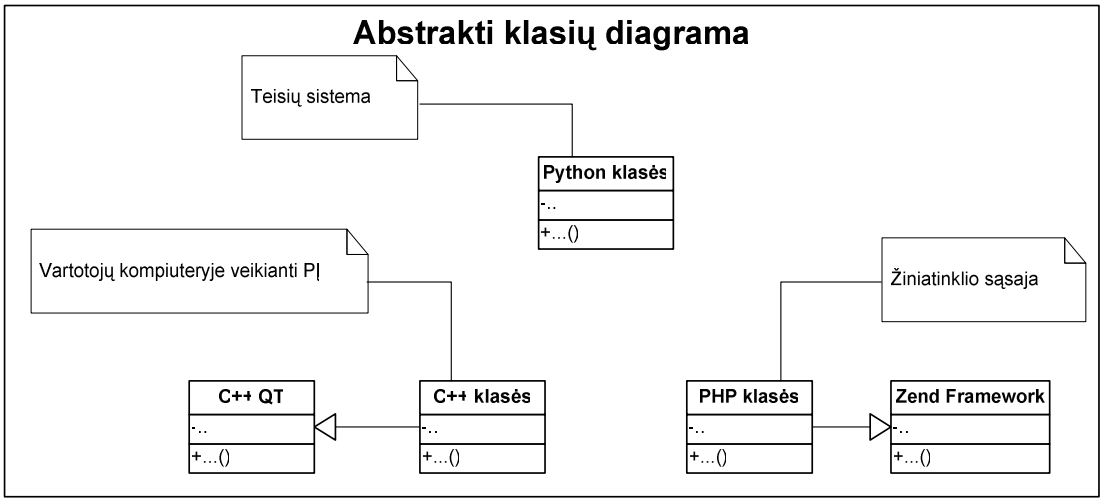
Vaidmenys gali būti priskiriami grupėms, vienas vaidmuo gali būti priskirtas kelioms grupėms. Kiekvieną kartą nebūtina kurti atskiro vaidmens kiekvienai grupei. Vaidmenys gali būti kuriami jei netinka nei vienas anksčiau aprašytas vaidmuo. Kiekvieną grupę sudaro vienas ar keli vartotojai , priskirti šiai grupei. Vartotojas gali priklausyti kelioms grupėms iš karto. Realizuota teisių sistema įgalina labai lengvai prijungti naujus modulius, sudaryti kompleksines teisių sąveikos kombinacijas.

3.4. Klasių diagrama

Sukurtą prototipą sudaro trys pagrindinės klasių grupės:

- PHP klasių grupė
- Python klasių grupė
- C++ klasių grupė

Tarpusavio sąveikia parodyta abstrakčiame klasių modelyje.



11 pav. Abstrakti klasių diagrama

Python klasės realizuotos DBVS lygyje yra atsakingos už teisių sistemą. Kiekvienas vartotojas jungiasi tiesiai prie duomenų bazės, bet mato skirtingus įrašus. Ši savybė realizuojama pasitelkus vaizdų („views“) kūrimo galimybes duomenų bazėje. Tokiu būdu buvo galima išvengti teisių realizavimo tiek vartotojų kompiuteryje veikiančioje programinėje įrangoje, tiek žiniatinklio sąsajoje.

C++ posistemės klasės paremtos C++ QT karkasu. Šis karkasas pasižymi tuo, kad galima kurti programinę įrangą įvairioms platformoms ir turi daug paruoštų komponentų vartotojo sąsajos kūrimui.

Žiniatinklio aplikacijos sluoksnis remiasi „Zend Framework“ karkasu. Šio karkaso naudojimas žymiai pagreitina programinės įrangos kūrimo procesą.

3.5. Testavimas

3.5.1. Testavimo tikslai ir apimtis

Testavimo tikslas – ištestuoti visus PĮ komponentus, jų tarpusavio sąveiką. Testavimu siekiama užtikrinti kuo aukštesnę PĮ kokybę. Tikrinamas PĮ atitikimas funkciniams ir nefunkciniams reikalavimams. Testavimo komanda stengsis aprėpti kiek įmanomą platesnį programinės įrangos funkcijų spektrą. Testavimo tikslai išskaidomi į tris potikslius:

- Žiniatinklio sąsajos testavimą.
- Vartotojų kompiuteryje veikiančios PĮ testavimą.
- Duomenų bazėje veikiančios PĮ testavimą. (PlPython).

Testavimo komandai būtina turėti testavimo planą siekiant ištestuoti kiek įmanoma daugiau PĮ komponenčių. Pagrindinis testavimo tikslas – užtikrinti pakankamą PĮ kokybę ir atitikimą reikalavimų specifikacijai.

Vienetų testavimas apims šias sistemos komponentes:

- PostgreSQL duomenų bazėje realizuotų komponenčių testavimą.
- Vartotojų kompiuteryje veikiančios sistemos komponenčių testavimą.
- Žiniatinklyje veikiančią programinės įrangos komponenčių testavimą.

Vienetų testavimas bus atliekamas baltos dėžės principu.

Integravimo testavimas apims šias PĮ dalis:

- Žiniatinklio modulių integraciją į visumą.

- QT Kliento modulių integraciją.
- PostgreSQL modulių integraciją.
- Visos sistemos integraciją į visumą.

Sistemos testavimas

- Žiniatinklio sąsajos testavimas ir atitikimas funkciniams reikalavimams.
- QT Kliento testavimas ir atitikimas funkciniams reikalavimams.

Priėmimo testavimas

- Bus žiūrima ar vartotojai turi pastabų veikiančiai PĮ, Visos pastabos bus fiksuojamos ir teikiami pasiūlymai PĮ tobulinimui.

Aukšto lygio testavimas

- Testavimo metu PĮ bus įdiegta vartotojų kompiuteriuose ir perkėlinėjami visi projektai iš dabartinės sistemos OpenCRM, žiūrima ar tenkinami projektų vadovų ir programuotojų poreikiai.

3.5.2. Pagrindiniai apribojimai

Priėmimo testavimas bus atliekamas „Coral solutions“ bendrovėje.

Baltos dėžės testavimas QT aplinkoje gali būti atliekamas tik iš kompiuterio, kuriame yra įdiegtas šis paketas. Reikalinga QT 4.3 versija ir „Visual Studio“.

Serverio pusės testavimas bus atliekamas tik kompiuteryje, kuriame įdiegtas „PostgreSQL“ duomenų bazės serveris su PlPython programinės įrangos paketu.

3.5.3. Vartotojo sąsajų testavimas

Programinę įrangą sudaro dvi pagrindinės sąsajos – tai žiniatinklio sąsaja, skirta klientams/išoriniams resursams ir QT kliento sąsaja, skirta darbui įmonės viduje. Kiekvienai iš šių sąsajų testavimas bus atliekamas atskirai.

3.5.3.1. QT Sąsajos testavimas

Vartotojų kompiuteryje veikiantis kliento funkcionalumas bus testuojamas šių programos langų atžvilgiu:

- Prisijungimas prie QT Kliento langas

- Teisių valdymo langas
- Vartotojo sukūrimo langas
- Klientų valdymo langas
- Konfigūracijos valdymo langas
- Projektų paieškos langas
- Ataskaitų generavimo langas
- Projekto langas
- Projekto iteracijos langas
- Darbo komentarų langas

3.5.3.2. Žiniatinklio sąsajos testavimas

Administratoriaus žiniatinklio sąsajos testavimas

- Kompanijų sąrašo langas
- Kompanijos redagavimo langas
- Kompanijos limitų nustatymo langas

Testuojama vartotojo žiniatinklio sąsaja:

- Prieinamų projektų sąrašas
- Naujos iteracijos sukūrimo langas
- Iteracijos redagavimo langas
- Darbo redagavimo sąsajos testavimas

3.5.3.3. Vienetų testavimas

Vienetų testavimas bus atliekamas baltos dėžės principu. Testavimas bus atliekamas pasitelkiant tris testavimo karkasus:

1. QxCPPUnit – CPP testavimui
2. PyUnit – plPython parašytoms procedūros ir klasėms testuoti.
3. PHPUnit – PHP kalba parašytoms programos dalims testuoti.

Vienetų testavimą atliks programuotojas, kuris realizavo atitinkamas sistemos komponentes.

3.5.3.4. Integravimo testavimas

Integravimo metu testuosime, kaip veikia sistemos dalys kartu. Pagrindinis uždavinys integravimo metu – užtikrinti, kad sąsaja tarp DBVS ir žiniatinklio, QT kliento veiktų korektiškai.

QT Kliento testavimo metu svarbu užtikrinti, kad korektiškai veiktų šie pagrindiniai PĮ komponentai:

- Vartotojo sąsaja
- Teisių valdymo posistemė
- Projektų valdymo posistemė
    i. Projekto posistemė
    ii. Iteracijos
    iii. Darbo
- Ataskaitų posistemė

Žiniatinklio sąsajos testavimo metu būtina užtikrinti šių modulių darnų darbą

- Teisių posistemės
- Vartotojų identifikacijos
- Projektų
- Iteracijos
- Darbo

Galutinio integravimo metu sujungiant modulius į vientisą sistemą bus naudojami kamščiai. Testavimas bus atliekamas smulkinančio testavimo pagrindu. Nedideli komponentai bus testuojami „Didžiojo sprogimo principu“, kai kelios klasės sujungiamos į programinį vienetą ir testuojamas visos iš karto. Testavimas remsis juodos dėžės principu. Bus žinomi tik grąžinami komponentų rezultatai

3.5.3.5. Testavimo rezultatai ir išvados

Testavimo metu buvo panaudoti du testavimo karkasai:

- CPPUnit – CPP klasių teavimui. CPP klasių testavimui buvo panaudotas ir papildomas QxRunner įrankis, skirtas vizualizuoti atliekamų testavimų rezultatus.
- PyUnit – „Python“ klasių testavimui/

Visos užfiksuotos klaidos buvo užregistruotos „Trac“ programinėje įrangoje.

Testavimo metu rastos klaidos teisių sistemoje buvo ištaisytos, o parašyti testiniai atvejai, padės išvengti šių klaidų pasikartojimo.

Atlikus testavimą programinė įranga veikė korektiškai Windows XP ir Windows Vista operacinėse sistemose.

Galime daryti išvadą, kad programinė įranga yra tinkama naudojimui.

3.6. Planavimo etapo charakteristikų fiksavimas sukurta sistema

Analitinėje dalyje buvo išskirtos šios keturios ekstremalaus programavimo planavimo etapo savybės:

- Idealus laikas
- Projekto greitis
- Vartotojo pasakojimai
- Užimtumo koeficientas

Sekančiuose skyreliuose detalizuosime kaip kiekvienas iš parametrų gali būti fiksuojamas sukurtoje programinėje įrangoje.

3.6.1. Idealaus laiko charakteristikos fiksavimas sukurta sistema

Analitinėje dalyje idealus laikas buvo apibrėžtas kaip laikas, praleistas prie konkretaus darbo be trukdymų. XPSuite sistemoje ši metrika buvo fiksuojama sekant programos langą, su kuriuo dirbama prie konkretaus darbo, bet galime įžvelgti šiuos šio metodo trūkumus:

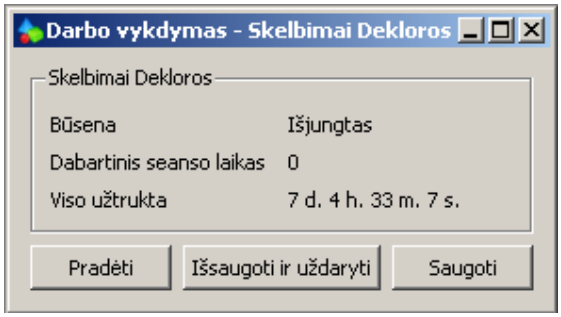
- Laiko sekimas buvo realizuotas tik konkrečioje integruoto kūrimo aplinkoje, įskiepio pagalba.
- Neskaičiuojamas užimtumo koeficientas.
- Kuriant komplikuotas sistemas dirbama ne su viena programa, o su keliomis programomis iš karto, Pvz.: interneto naršyklė, „Zend Studio“.
- Sukurtas įrankis remiasi tik planavimo etapo idealaus laiko fiksavimo galimybėmis.
- Sukurto įrankio pagalba negalimas kelių programuotojų darbo laiko sumavimas. Įrankis skirtas naudoti vieno programuotojo atžvilgiu.

Sukurtoje sistemoje šį parametrą buvo nuspręsta fiksuoti kitu principu. Šiam tikslui buvo sukurtas atskiras prototipas, nesusijęs su jokiu naudojamu programinės įrangos paketu. Prototipe sukurtas atskiras langas, kurio pagalba galima lengvai perjunginėti darbus, prie kurių konkrečiu laiko momentu dirbama. Vienu metu gali būti atidaryti keli laiko fiksavimo

langai, tai leidžia greitai perjunginėti darbus. Realizuotame lange realizuotos trys paprastos funkcijos:

- Pradėti/sustabdyti darbą
- Saugoti
- Išsaugoti ir uždaryti

Šių trijų funkcijų pagalba valdomas visas idealaus laiko sekimas. Lango realizacija pateikta 11 paveikslėlyje.



12 pav. Idealaus laiko fiksavimo langas

Visa sukaupta idealaus laiko statistika pateikiama vartotojų ataskaitoje. Ataskaitos dalies pavyzdys pateiktas 13 paveikslėlyje. Pilnas ataskaitos vaizdas pateikiamas 9.1 priedo punkte.

Skelbimai	**0 Lt.**	**0 s.**				
Skelbimai Dekloros	0 Lt.	20 d.	4 s.	7.56899 d.	2009-04-19 19:10:06	
Skelbimai Dekloros	0 Lt.	20 d.	7.7075 h.	7.56899 d.	2009-04-20 16:59:31	
Skelbimai Dekloros	0 Lt.	20 d.	7.00667 h.	7.56899 d.	2009-04-21 16:59:35	
Skelbimai Dekloros	0 Lt.	20 d.	6.28361 h.	7.56899 d.	2009-04-23 16:32:52	
Skelbimai Dekloros	0 Lt.	20 d.	5.24556 h.	7.56899 d.	2009-04-24 15:57:24	
					Sugaištas laikas:	**3.28056 d.**
					Biudžetas:	**0.0 Lt.**

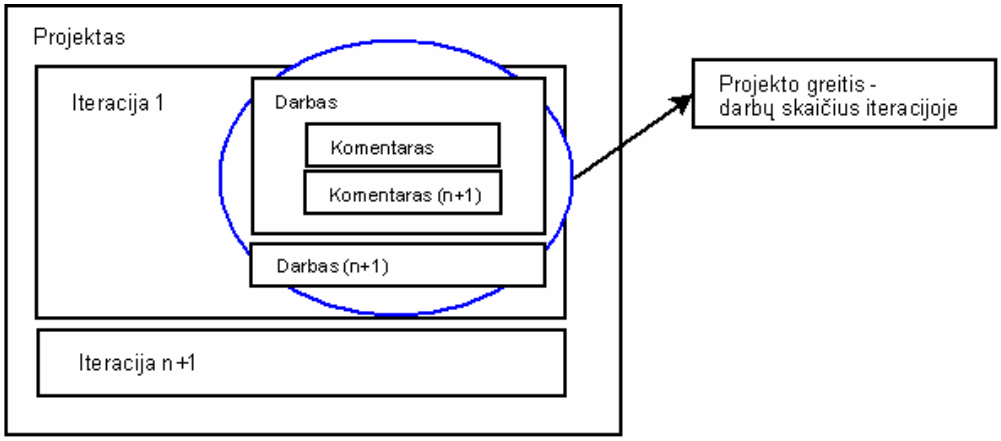
13 pav. Ataskaitos žurnalo dalies pavyzdys

3.6.2. Projekto greičio metrikos vertinimas

Projekto greičio charakteristikai įvertinti pasiremsime apibrėžimu, kuriame sakoma, kad projekto greitis yra realizuotų vartotojo pasakojimų skaičius iteracijoje. Sukurtoje programinėje įrangoje realizuotas iteracijos esybę realizuojantis programos langas. Iteracijos lango ekrano nuotrauka pateikiama priede. Iteracijos lange fiksuojamas vartotojo pasakojimas, kuris gali turėti atskiras versijas. Kiekvienas darbas priskiriamas atitinkamos versijos iteracijai. Šiuo atveju projekto greitaveiką galima laikyti vartotojo pasakojimų skaičių

iteracijoje. Jeigu kiekvienas darbas atspindi atskirą vartotojo pasakojimo istoriją, tai juos galime laikyti projekto greitaveika.

Projekto greitaveiką galima skaičiuoti remiantis 12 pav. Siekiant paprastumo eksperimento metu, detalūs reikalavimai nebuvo skaidomi į atskirus darbus, o viskas surašoma iteracijoje. Projekto greitis buvo skaičiuojamas rankiniu būdu.



14 pav. Projekto greičio vertinimas

3.6.3. Vartotojo pasakojimų valdymas

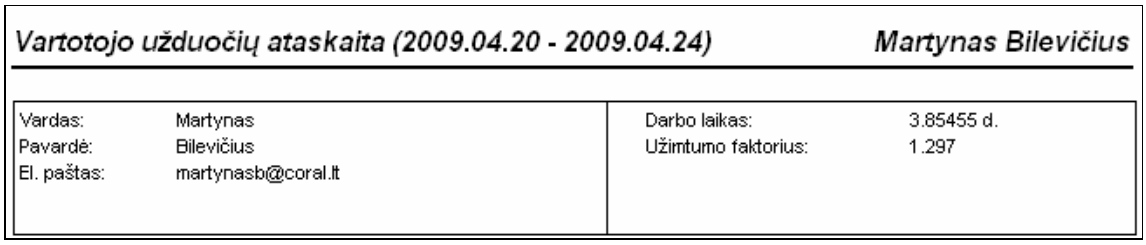
Vartotojų pasakojimai kaupiami iteracijos lygmenyje. Pasakojimai sukurtoje sistemoje gali turėti skirtingas versijas. Kiekvienas darbas yra susiejamas su konkrečia iteracija ir vartotojo pasakojimo versijos numeriu. Vartotojo pasakojimas gali būti atnaujintas iš dviejų šaltinių:

- Žiniatinklio posistemės
- Vartotojų kompiuteryje veikiančios sistemos.

Jei iteracijos pasakojimas atnaujinamas iš žiniatinklio posistemės, tai automatiškai padidinama vartotojo versija. Jei atnaujinimas vyksta iš darbuotojo kompiuteryje veikiančios sistemos, jis gali pasirinkti ar tiesiog pataisyti pasakojimą, ar pataisyti ir publikuoti kaip naują versiją.

3.6.4. Užimtumo koeficiento skaičiavimas

Kadangi idealus laikas yra fiksuojamas programoje, galime apskaičiuoti ir užimtumo koeficientą. Užimtumo koeficientas skaičiuojamas automatiškai vartotojų užimtumo žurnale.



15 pav. Ataskaitos žurnalo apibendrinti duomenys

Detalūs skaičiavimai remiantis surinktais duomenimis bus pateikti empiriniame tyrime.

4. PROGRAMINĖS ĮRANGOS DIEGIMO TYRIMAS

Tyrimo dalyje įvertinami kokybiniai sukurtos programinės įrangos parametrai. Įvertinami vartotojų pateikti programinės įrangos tobulinimo pasiūlymai.

4.1. Atlikto darbo kokybės analizės tikslai

Įvertinti šiuos sukurtos programinės įrangos parametrus:

- Aptikti klaidas funkcionavime, logikoje, realizacijoje
- Patikrinti ar programinė įranga atitinka reikalavimų specifikaciją
- Įvertinti vartotojų atsiliepimus naudojantis sistemą
- Įvertinti programinės įrangos tobulinimo galimybes

4.2. Sistemos funkcionalumo tyrimas

Sistemos funkcionalumo tyrime dalyvavo šie programinės įrangos naudotojai:

- Programuotojai
- Dizaineriai
- Projektų vadovai

Programine įranga jau naudojamasi apie keturis mėnesius. Eksperimento pabaigoje buvo prašoma programa naudojusių darbuotojų išsakyti savo pastebėjimus. Dauguma vartotojų išsakytų reikalavimų nebuvo aprašyti reikalavimų specifikacijoje, todėl jie bus įgyvendinti vėlesnėse PĮ vystymo etapuose.

4 lentelė. Programuotojų išsakyti reikalavimai

Reikalavimo Nr.	**Aprašymas**
1.	Programa turi startuoti automatiškai su operacine sistema.
2.	Įjungta programa turi prisiminti prieš tai buvusios sesijos atidarytus langus.

3.	Uždarant programą turi būti automatiškai išsaugomas užgaištas laikas.
4	Fiksuojant idealų laiką leisti įvesti pastabas.

Įvertinant programuotojų išsakytus reikalavimus pastebime, kad išsakyti reikalavimai daugiausia yra susiję su pastoviu programos naudojimosi. Šiuo metu buvo įgyvendintas trečiu numeriu pažymėtas reikalavimas.

5 lentelė. Dizainerio išsakyti reikalavimai

Reikalavimo Nr.	Aprašymas
1.	Programa turi fiksuoti įjungimo ir darbo saugojimo veiksmus žurnale. Dizaineris vienu metu atlieka daug darbų ir sunku susigaudyti kada pradėtas ir kiek truko konkretus darbas, kol jis nėra baigtas.

Dizainerio išsakytas reikalavimas glaudžiai siejasi su jo darbo specifika. Dienos eigoje dizaineris atlieka labai daug smulkių darbų. Atliekant daug smulkių ir mažos trukmės darbų, sunku mėnesio pabaigoje pateikti darbų ataskaitą, kadangi daugelis darbų nebūna baigti ir kol darbas nėra baigtas, jis neatsispindi bendroje ataskaitoje. Šiai problemai spręsti buvo pasiūlyta įdiegti žurnalo funkciją, kuri fiksuotų kada ir koks darbas buvo daromas ir kiek truko. Šios funkcijos įgyvendinimas leido tiksliai pateikti darbų ataskaitą, kadangi galima sužinoti kada ir koks darbas buvo darytas neužbaigiant jo. Šios funkcijos pagalba realizuojamas ir idealaus laiko fiksavimo algoritmas.

Naudojantis žurnalo funkcija buvo pastebėta, kad norint neprarasti su programų priežiūra susijusių darbų, iškilo dar vienas reikalavimas. Prieš generuojant ataskaitą turėtų būtų leidžiama pasirinkti, kokie darbai turėtų būti įtraukiami į idealų laiką, o kurie turėtų būti laikomi priežiūros darbais.

Pastebėta, kad galima realizuoti darbų pabaigos prognozavimo funkciją, kuri remtųsi prognozuojama darbų trukme ir užimtumo koeficientu. Darbų trukmė būtų apskaičiuojama sudauginant likusį laiką darbui pabaigti ir užimtumo koeficientą.

6 lentelė. Projekto vadovo išsakyti reikalavimai

Reikalavimo Nr.	Aprašymas
1.	Dabartinis projektų pateikimas turi būti pakeistas išsišakojančiu. Jis įgalintų lengvesnį projekto koordinavimą ir reikiamos informacijos pasiekimą.

Projekto vadovo išsakytas reikalavimas žymiai pagerintų navigaciją projekto lygyje. Jo reikalavimas leistų daug vizualiau perteikti sistemos koncepcinį modelį. Išsakytas reikalavimas bus įgyvendintas tolimesnėse programinės įrangos vystymo stadijose.

4.3. Tyrimo išvados

Atlikus tyrimą buvo gauta naudingos informacijos iš vartotojų. Pastebime, kad kiekvienas vartotojas akcentuoja jam labiausiai rūpimas programinės įrangos savybes. Kadangi buvo pasirinkta daugelį esminių funkcijų realizuoti duomenų bazės lygyje, buvo nesunku praplėsti sistemos savybes naujais išsakytais dizainerio reikalavimais.

Taip pat pastebime, kad dauguma išsakytų reikalavimų siejasi su sistemos panaudojamumo savybėmis. Reikia pastabėti, kad pasirinkta architektūra ypač lengvai leidžia realizuoti naujus funkcinius reikalavimus. Pasirinktas „QT“ programavimo karkasas leidžia nesunkiai praplėsti vartotojų išsakytus panaudojamumo reikalavimus. Gauti vertingi atsiliepimai, kaip būtų galima papildyti planavimo etapo charakteristikų stebėjimo būdus.

5. PLANAVIMO ETAPO EMPIRINIS TYRIMAS

5.1. Tikslas

Empiriniu metodu ištirti planavimo etapo charakteristikas ir didžiausias įtakas daromas šio etapo pirminiams laiko vertinimams. Empiriškai įvertinti aprašytas ekstremalaus programavimo planavimo etapo charakteristikas. Eksperimento pasirinkimą lėmė iki šiol buvęs netikslus projektų trukmės prognozavimas. Nebuvo statistikos, kurios pagalba būtų galima tiksliau įvertinti planuojamų darbų trukmes. Daugelis darbų trukdavo ilgiau nei būdavo planuojama. Tikimasi, kad sukaupti eksperimento rezultatai leis tiksliau įvertinti prognozuojamų darbų trukmes ateityje., pateiks vertingos statistikos, susijusios su planavimo etapu.

5.2. Eksperimento objektas

Tyrimas remiasi planavimo etapo keturiomis charakteristikomis:

- Idealus laikas
- Projekto greitis
- Vartotojų pasakojimai
- Užimtumo koeficientas

5.3. Eksperimento dalyviai

Eksperimente dalyvaus du pagrindiniai įmonės darbuotojai, atliekantys programavimo darbus. Darbuotojai sukurtos programinės įrangos pagalba fiksuos atliekamus darbus. Tyrimo

pabaigoje bus įvertinti surinkti duomenys. Eksperimentas bus atliekamas „Coral Solutions“ įmonėje. Eksperimentinėje dalyje pateikti tikri duomenys surinkti empirinio tyrimo metu.

5.4. Eksperimento priemonės

Eksperimentas bus atliekamas sukurtos programinės įrangos pagalba. Programinėje įrangoje prieš atliekant eksperimentą jau buvo realizuota idealaus laiko fiksavimo funkcija.

5.5. Eksperimento eiga

5.5.1. Pirmo programuotojo surinkti duomenys:

Pirmasis programuotojas naudojosi „Windows Vista“ operacine sistema. Darbuotojas fiksavo darbus keturias dienas per savaitę.

Pirmam darbui aprašyti pasirinkta lentelė.

7 lentelė. Pirmo darbo duomenys

Pavadinimas	**Planuojama trukmė**
Planuojama trukmė	60h
Idealus laikas	62h
Prognozavimo tikslumas	94 %
Pradžios data	2008-03-23
Pabaigos data	2008-04-13
Kalendorinis laikas	90h
Užimtumo koeficientas	1,45

Kalendorinės darbo valandos buvo apskaičiuotos remiantis darbuotojo darbo grafiku. Savaitės laikotarpyje buvo dirbama 4 dienas, viso 90h. Kas pažymėtina apie atliktą darbą, kad dar nebuvo atliktas vartotojo testavimas, kas gali įtakoti idealų darbą. Visą darbą sudarė apie 35 vartotojo reikalavimai susiję su šiuo darbu. Juos galime laikyti projekto greitaveika.

8 lentelė. Antro darbo duomenys

Pavadinimas	**Planuojama trukmė**
Planuojama trukmė	160h
Idealus laikas	61h
Prognozavimo tikslumas	-
Pradžios data	2008-04-14
Pabaigos data	2008-05-02
Maksimalus laikas	80h
Užimtumo koeficientas	1,31

Kadangi antras darbas dar nėra baigtas, galime apskaičiuoti planuojamo darbo trukmę, remiantis užimtumo koeficientu. Šiuo atveju, apskaičiuodami būsimą darbų trukmę, galėtume remtis priede pateiktu automatiškai apskaičiuojamu užimtumo koeficientu. Šiam tikslui turėtų būti patobulinta idealaus laiko skaičiavimo ataskaitos generavimo forma, kad būtų galima pasirinkti dirbamas dienas ir darbus, kurių atžvilgiu turėtu būti skaičiuojamas šis koeficientas. Dėl pastarosios priežasties skaičiavimas buvo atliekamas rankiniu būdų, pasinaudojus idealiu laiku. Antrojo darbo greitaveiką remiantis pateiktais duomenimis sudaro 34 vartotojo reikalavimai.. Pastebime, kad užgaištas laikas ir vartotojo išreikštų reikalavimų realizacijos laikas beveik sutampa su pirmojo darbo rezultatais. Tai patvirtina, kad galime remtis anksčiau darytų darbų rezultatais ir taikyti „vakarykščio oro“ metodą skaičiuojant darbų trukmę.

Antrąjį darbą sudarė du moduliai:

- Skelbimų modulis, kurio trukmė ir pateikta 8 lentelėje
- Filtro modulis, kuris dar yra kuriamas

Apskaičiuojame kiek kalendorinių darbo valandų reikės.
(160-61),.31 = 129h arba 16,2 darbo dienos.*

Jei laikysimės šio skaičiavimo ir prognozuojama darbų trukmė sutaps su sugaištu laiku tai – antras modulis turėtų būti baigtas gegužės mėnesio pirmomis dienomis. Šitoks skaičiavimo metodas dar vadinamas „vakarykščio oro“ metodu..

Apskaičiuokime bendrą užimtumo koeficientą viso stebėjimo laikotarpiui:
(170/123) = 1,38

Pastebime, kad užimtumo koeficientas sudaro beveik 40% procentų viso laiko. Galime daryti išvadas, kad užimtumo koeficientas daro didžiausią įtaką prognozuojamų darbų pabaigos terminams. Nereikia pamiršti, kad stebėti darbai negalėjo remtis ankstesnių darbų prognozėmis, kadangi jie nebuvo anksčiau daryti. Todėl galime numatyti, kad paklaida bus dar didesnė.

Tolimesnio stebėjimo rezultatai tik patvirtino pirminio stebėjimo išvadą, kad didžiausią įtaką darbų trukmei turi priežiūros darbai. Gegužės 4-7 dienų atliktais stebėjimo duomenimis (16. pav) užimtumo koeficientas buvo apie vienetą. Šią savaitę beveik nebuvo trukdymų, todėl efektyviai buvo išnaudotas kalendorinis laikas. Tačiau jau kitą savaitę, šis koeficientas šoktelėjo iki 45 % ir vidutinis užimtumo koeficientas vėl stabilizavosi.

Vartotojo užduočių ataskaita (2009.05.04 - 2009.05.07) *Remigijus Kiminas*

Vardas:	Remigijus	Darbo laikas:	3.81472 d.
Pavardė:	Kiminas	Užimtumo faktorius:	1.049
El. paštas:	remigijusk@coral.lt		

16 pav. Gegužės 4-9 darbų žurnalo statistika

5.5.2. Antrojo programuotojo surinkti duomenys

Prieš apibendrindami antrojo programuotojo sukauptus duomenis, pateikime antrojo programuotojo surinktus duomenis Balandžio 20-24 dienomis. Penkių dienų žurnalas pateiktas 17 paveikslėlyje:

Vartotojo užduočių ataskaita (2009.04.20 - 2009.04.24) *Martynas Bilevičius*

Vardas:	Martynas	Darbo laikas:	3.85455 d.
Pavardė:	Bilevičius	Užimtumo faktorius:	1.297
El. paštas:	martynasb@coral.lt		

DC	**0 Lt.**	**0 s.**				
Registrarshel - ENSIM	0 Lt.	0 s.	19.1667 min.	3.96115 d.	2009-04-20 16:48:04	
Registrarshel - ENSIM	0 Lt.	0 s.	6.21222 h.	3.96115 d.	2009-04-21 13:22:26	
Kiti darbai	0 Lt.	0 s.	15.7 min.	15.7 min.	2009-04-23 16:19:27	
Registrarshel - ENSIM	0 Lt.	0 s.	4.11111 h.	3.96115 d.	2009-04-23 13:20:12	
Registrarshel - ENSIM	0 Lt.	0 s.	5.525 h.	3.96115 d.	2009-04-24 13:28:08	
					Sugaištas laikas:	**2.05368 d.**
					Biudžetas:	**0.0 Lt.**
Autoera	**0 Lt.**	**0 s.**				
Autoera shop	0 Lt.	0 s.	2.44806 h.	5 d.	2009-04-21 16:41:48	
Autoera shop	0 Lt.	0 s.	2.01111 h.	5 d.	2009-04-22 16:51:27	
Autoera shop	0 Lt.	0 s.	5.74222 h.	5 d.	2009-04-23 16:59:53	
Autoera shop	0 Lt.	0 s.	4.20556 h.	5 d.	2009-04-24 15:48:39	
					Sugaištas laikas:	**1.80087 d.**
					Biudžetas:	**0.0 Lt.**

17 pav. Vartotojo darbų žurnalas

Kaip matyti iš surinktų duomenų, užimtumo koeficientas buvo apskaičiuotas automatiškai ir sudarė 1.29 punkto. Procentine išraiška būtų apie 30%. Antras programuotojas nevedė prognozuojamų laikų, todėl šis koeficientas nebuvo skaičiuojamas. Iš automatiškai suskaičiuoto užimtumo koeficiento matome, kad jis labai artimas pirmojo programuotojo užimtumo koeficientui.

5.6. Empirinio tyrimo išvados

Atlikus empirinį tyrimą buvo nustatyta, kad didžiausią įtaką projekto trukmei turi priežiūros darbai. Šiuo atveju, priežiūra laikoma visi papildomi darbai, kurie nėra susiję su konkrečiu metu atliekamu pagrindiniu darbu. Daugiausia priežiūros didėjimą lemią su PĮ įrangos palaikymu susijusios užduotys.

Prognozavimo tikslumas buvo ganėtinai aukštas, tačiau reikia pastebėti, kad ankstesniais matavimais, darbo prognozavimo tikslumo paklaida siekė 30%. Darbo trukmės įvertinimo tikslumas priklauso nuo programuotojo patirties ir ar anksčiau buvo atliekamas panašus darbas.

Norint gauti tikslius duomenis kurie ateityje atneštų naudos, svarbią vietą užima ir socialinis faktorius. Jeigu vartotojai nenorės naudotis PĮ ir detaliai fiksuoti atliekamų darbų, gauti rezultatai bus netikslūs.

Eksperimento metu paaiškėjo, jog norėdami neprarasti informacijos apie pasirinktą savaitę darytus darbus ir tiksliai apskaičiuoti užimtumo koeficientą, būtini tokie PĮ patobulinimai.

- Turi būti sukurta galimybė pažymėti darbo dienas prieš generuojant ataskaitą
- Turi būti sukurta galimybė pažymėti darbus, kurių atžvilgiu bus skaičiuojamas užimtumo koeficientas
- Turi būti sukurta galimybė sudaryti savo darbo kalendorių. Nurodyti šventines dienas, darbo valandas, atostogas.

Buvo nustatyta, kad abiejų programuotojų užimtumo koeficientas siekia apie 30%. Anksčiau įmonės viduje nebuvo žinoma, kiek užima priežiūra laiko atžvilgiu, todėl gauti rezultatai bus naudingi planuojant tikslesnius darbų terminus. Kadangi užimtumo koeficientas skaičiuojamas automatiškai, jį galima panaudoti paskirstant priežiūros darbus savaitės laikotarpyje po lygiai visiems programuotojams, tokiu būdu galima valdyti idealų laiką skirtą kiekvienam programuotojui.

Kai žinomas užimtumo koeficientas ir jis stabilizuojas turi būti susitarta dėl:

- Kokius laikus pateikia programuotojas (su įvertintu užimtumo koeficientu ar be jo).
- Kaip bus valdomas prognozavimo tikslumas.

Iš programinės įrangos panaudojamumo pusės buvo pastebėta, kad dienos eigoje pakankamai aktyviai reikia perjunginėti darbus, todėl ateityje būtų galima įdiegti greitų klavišų kombinacijas. Kartais užmirštama išjungti aktyvius darbus. Šiai problemai spręsti ateityje būtų galima įdiegti ir vartotojo aktyvumo fiksavimą dviem būdais:

- Įvesti galimybę sustabdyti laikmatį automatiškai atsižvelgiant į klaviatūros mygtukų spaudimų dažnumą. Jei nespaudžiamas nei vienas mygtukas penkių minučių laikotarpyje, laikmatis sustabdomas automatiškai.
- Fiksuoti kursoriaus aktyvumą tam tikrame intervale. Jei kursorius neaktyvus daugiau kaip dešimt minučių, automatiškai sustabdyti atliekamą darbą.

Tai palengvintų automatinį idealaus laiko skaičiavimą. Taikant šias savybes su įgyvendintom darbo kalendoriaus savybėmis, būtų galima iki minimumo sumažinti reikalingą laiką perjunginėjant darbus. Galbūt būtų galima apskaičiuoti ir kitokią statistiką. Daugelis darbo trukmės/sudėtingumo modelių siūlo vertinti programos sudėtingumą kodo eilučių skaičiumi. Būtų galima pateikti idealaus laiko ir klavišų paspaudimo dažnio statistiką.

Pakankamai tikslūs duomenys gaunami taikant „vakarykščio oro“ metodą. Darbus vertinant anksčiau atliktų darbų atžvilgiu. Taikant šį metodą reikia įvertinti:

- Ar tolimesnius darbus darys tas pats programuotojas
- Įvertinti naujos vartotojo istorijos sudėtingumo lygį
- Ar reikalavimus pateikė tas pats vartotojas

Buvo pastebėta, kad pakankamai sunku atskirti konkrečiam darbui skiriamą darbą, jei kuriamos sistemos moduliai glaudžiai siejasi vienas su kitu. Šiai problemai spręsti būtų galima įvesti trečią darbą, kuris būtų naudojamas trukmei fiksuoti, kuriant sąveikaujančias modulių dalis.

Atliktas empirinis tyrimas atskleidė, kad:

- Prognozavimo tikslumui gerinti reikia naudoti „vakarykščio oro“ metodą.
- Užimtumo koeficientas daro didžiausią įtaką prognozuojamų darbų pabaigos trukmei.
- Užimtumo koeficientas yra sąlyginai vienodas abiem programuotojams dirbantiems įmonėje.

6. IŠVADOS

- Sukurtas ekstremaliu programavimu pagrįstos projektų valdymo sistemos prototipas.
- Realizuotos visos esminės ekstremalaus programavimo metodikoje egzistuojančios programinės kūrimo proceso dalys:
    - Projektas
    - Iteracija

    - Darbas
- Sukurta galimybė realiu laiku fiksuoti darbų trukmę, kas žymiai supaprastino idealų darbo trukmės fiksavimą. Automatiškai apskaičiuojamas užimtumo koeficientas ir idealus laikas. Remiantis pastaraisiais parametrais galima daug tiksliau prognozuoti darbų trukmes.
- Nustatyta, kad empirinio tyrimo metu įmonėje vidutinė darbo trukmės prognozavimo tikslumas siekė 70%, vidutinis užimtumo koeficientas 1,3.
- Daug tikslesnius vertinimus pateikia programuotojai, turintys didesnę darbo patirtį.
- Eksperimento metu buvo nustatyta, kad didžiausią įtaka pirminiams laiko vertinimams turi PĮ priežiūros darbai, kurie savaitės laikotarpyje sudaro iki 40% viso sugaišto laiko.
- Visiškas ekstremalaus programavimo metodikos įsisavinimas dabartinėje situacijoje neįmanomas nes, klientai visada nori žinoti galutinę kainą pirminės stadijos metu, prieš pasirašant sutartį.
- Siekiant kuo tiksliau įvertinti planuojamų darbų trukmes būtina įvertinti:
    - Kiekvieno programuotojo vidutinį darbo trukmės prognozavimo tikslumą.
    - Kiekvieno programuotojo užimtumo koeficientą atlikus stebėjimus.
    - Atliktų darbų kaupimas leidžia pasinaudoti „vakarykščio oro“ metodu.
- Didelę įtaką renkamos informacijos kokybei daro socialiniai faktoriai, ar vartotojas noriai naudojasi sukurta programine įranga ir ar detaliai fiksuoja atliekamus darbus. Būtina darbuotojų motyvacija naudotis programine įranga. Ypač svarbios tampa panaudojamumo savybės idealaus laiko fiksavimo metu.
- Jei įmonėje dirba panašaus lygio programuotojai, jų užimtumo koeficientas gali būti stabilizuotas paskirstant darbus priklausomai nuo savaitės laikotarpyje kintančio užimtumo koeficiento. Užimtumo koeficientas sukurtoje programinėje įrangoje gali būti apskaičiuojamas automatiškai.
- PĮ kūrimo proceso gerinimui gali būti panaudota pasirinkto programinės įrangos kūrimo modelio dalis. Nebūtina remtis visomis aprašytomis dalimis pasirinktoje metodikoje. Eksperimento metu įmonėje buvo pritaikytas ekstremalaus programavimo planavimo etapas.

7. LITERATŪRA

[1] JAMIE L. SMITH; SHAWN A. BOHNER; D. SCOTT MCCRICKARD. Project Management for the 21st Century: Supporting Collaborative Design through Risk Analysis. [Interaktyvus] 2005. 300p [Žiūrėta 2007 10 28]. Prieiga per ACM Digital library <http://portal.acm.org/citation.cfm?id=1167319&coll=Portal&dl=ACM&CFID=40716985&CFTOKEN=27552456>

[2] TAYLOR, H. The Move to Outsourced IT Projects: Key Risks from the Provider Perspective. [Interaktyvus] 2005. 149p. ISBN:1-59593-011-6 . Prieiga per ACM Digital library <http://portal.acm.org/citation.cfm?id=1056006&coll=Portal&dl=ACM&CFID=40716985&CFTOKEN=27552456>

[3] Extreme Chaos [Interaktyvus] 2001. [Žiūrėta 2007 10 28]. Prieiga per Internetą <http://www.vertexlogic.com/processOnline/processData/documents/pdf/extreme_chaos.pdf >

[4] M. RITA THISSEN; JEAN M. PAGE; MADHAVI C. BHARATHI; TOYIA L. AUSTIN. Communication tools for distributed software development teams [Interaktyvus] 2007. 28p. ISBN:978-1-59593-641-7 [Žiūrėta 2007 10 28]. Prieiga per ACM Digital library <http://portal.acm.org/citation.cfm?id=1235007&coll=Portal&dl=ACM&CFID=40716985&CFTOKEN=27552456 >

[5] GEORGE, S. Software project secrets why software projects fail. Apress, 2005. 113 p. ISBN – 1-59059-550-5.

[6] PRITCHARD, C. The Project Management Communications Toolkit. Artech House, 2004. 10 p. ISBN 1580537472.

[7] BITTNER, K.; SPENCE, I. Managing Iterative Software Development Projects. Addison Wesley Professional, 2006. 2 skyrius. ISBN-13: 978-0-321-26889-1.

[8] BIGELOW, D. Will Outsourcing Relationships Rule the New Millenium?. [Interaktyvus] 2002. Volume 16, Number 5 [Žiūrėta 2007-10-28]. Prieiga per Internetą <http://www.pmsolutions.com/uploads/pdfs/outsourcing_rule.pdf>

[9] SNEDAKER, S. How to Cheat at IT Project Management. Syngress, 2005. 1.4 skyrius. ISBN- 1-59749-037-7.

[10] Project management software [Žiūrėta 2007-10-28], prieiga per Internetą <http://en.wikipedia.org/wiki/Project_management_software>

[11] ZHANG, S.; ZHAO, C.;ZHANG, Q.; SU, H.; GUO, H.; CUI, J.; PAN, Y.; MOODY, P. Managing Collaborative Activities in Project Management. [Interaktyvus] 2007. 1-2p [Žiūrėta 2007 11 02]. Prieiga per ACM Digital library <http://portal.acm.org/citation.cfm?id=1234791&coll=Portal&dl=ACM&CFID=5004938&CFTOKEN=98110176>

[12] E MAQSOOD, M.;JAVED, T. Practicum in software project management: an endeavor to effective and pragmatic software project management education. [Interaktyvus] 2007. 573p [Žiūrėta 2007 11 02]. Prieiga per ACM Digital library

<http://portal.acm.org/citation.cfm?id=1287691&coll=Portal&dl=ACM&CFID=5004938&CFTOKEN=98110176>

[13] L. REID, K.; V. WILSON, G. DrProject: a software project management portal to meet educational needs. [Interaktyvus] 2007. 573p [Žiūrėta 2007 11 02]. Prieiga per ACM Digital library <http://portal.acm.org/citation.cfm?id=1227421&coll=Portal&dl=ACM&CFID=5004938&CFTOKEN=98110176 >

[14] ANGIONI, M.; CARBONI, D.; MELIS,M.; PINNA, S.; SANNA, R.; SORO, A. XPSuite: tracking and managing XP projects in the IDE. [Interaktyvus] 2004. 52p [Žiūrėta 2007 11 03]. Prieiga per ACM Digital library <http://portal.acm.org/citation.cfm?id=1151440&coll=Portal&dl=ACM&CFID=5094523&CFTOKEN=37873938 >

[15] CHARVAT, J. Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects. John Wiley & Sons, 2003. 4 skyrius. ISBN:0471221783.

[16] Microsoft Project Reviewer's Comments [Žiūrėta 2007-11-03], prieiga per Internetą <http://project-management-software-review.toptenreviews.com/microsoft-project-review.html>

[17] CUSUMANO, MICHAEL A. Extreme programming compared with Microsoft-style iterative development. [Interaktyvus] 2007. 15p [Žiūrėta 2007 11 03]. Prieiga per ACM Digital library <http://portal.acm.org/citation.cfm?id=1290979&coll=Portal&dl=ACM&CFID=5116660&CFTOKEN=88015818>

[18] Microsoft Project Reviewer's Comments [Žiūrėta 2007-11-03], prieiga per Internetą <http://project-management-software-review.toptenreviews.com/microsoft-project-review.html>

[19] BOOCH, G.; C, MARTIN, R.; NEWKIRK, J. Object Oriented Analysis and Design with Applications. Addison Wesley, 1998. 4 skyrius.[Žiūrėta 2007-11-03], prieiga per Internetą <http://www.objectmentor.com/resources/articles/RUPvsXP.pdf>

[20] CHARVAT, J. Building Web Applications with UML Second Edition. Addison Wesley, 2002. 7 skyrius. ISBN: 0-201-73038-3.

[21] DARIE, C.; BRINZAREA, B.;CHERECHEŞ-TOŞA, F.; BUCICA, M. AJAX and PHP: Building Responsive Web Applications. Packt Publishing Ltd., 2006. ISBN: 1904811825.

[22] PAUL D., S. AJAX and PHP: Building Responsive Web Applications. Packt Publishing Ltd., 2006. ISBN: 1904811825.

[23] ANDREWS M.; A. WHITTAKER, J. How to Break Web Software: Functional and Security Testing of Web. Addison-Wesley Professional, 2006. 10p. ISBN: 0321369440.

[24] OPPLIGER, R. Internet and Intranet Security, Second Edition. ARTECH HOUSE, 2002. 15 skyrius. ISBN: 1-58053-166-0.

[25] PEIKARI, C.; CHUVAKIN, A. Security Warrior. O'Reilly, 2004. 16.3 skyrius. ISBN: 0-596-00545-8.

[26] K. BOARDMAN, S.; CAFFREY, M.;MORSE, S. Oracle Web Application Programming for PL/SQL Developers. Prentice Hall PTR, 2002. 12p. ISBN: 0130477311.

[27] Introduction to PL/SQL [Interaktyvus] [Žiūrėta 2007 11 04], prieiga per Internetą <http://pd.acm.org/sks_course.cfm?crs=216518_eng>

[28] MCDONALD, C. Mastering Oracle PL/SQL: Practical Solutions. Springer, 2004. 4-6p. ISBN: 1590592174.

[29] DOUGLAS, K.; DOUGLAS, S. PostgreSQL. Sams Publishing, 2003. ISBN: 0-7357-1257-3.

[30] BLANCHETTE J.; SUMMERFIELD M., S. C++ GUI Programming with Qt 3. Prentice Hall PTR, 2004. ISBN: 0-13-124072-2.

[31] W. BOEHM, B.; TURNER, R. Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley Professional, 2003. ISBN: 0321186125.

[32] GROSSMAN, F.; BERGIN, J.; LEIP, D.; MERRITT, S.; GOTEL, O. One XP experience: introducing agile (XP) software development into a culture that is willing but not ready. [Interaktyvus] 2004. 15p [Žiūrėta 2007 11 10]. Prieiga per ACM Digital library <http://portal.acm.org/citation.cfm?id=1034933&coll=Portal&dl=ACM&CFID=5853907&CFTOKEN=20254661>

[33] KIRCHER, M.; JAIN, P.; CORSARO, A.;LEVINE, D. eXtreme Programming in Open-Source and Distributed Environments [Interaktyvus] [Žiūrėta 2007 11 07], prieiga per Internetą <http://www.kircher-schwanninger.de/michael/publications/DistributedXP.pdf>

[34] BECK, K. eXtreme Extreme Programming Explained. ISBN: 0201616416

[35] MILLER, R.; COLLINS , C. XP distiled. [Interaktyvus] [Žiūrėta 2009 05 01], prieiga per Internetą <http://www.ibm.com/developerworks/java/library/j-xp/>

[36] BECK, K.; FOWLER , M. Planning Extreme Programming (XP Series) (Paperback). ISBN 0201710919

[37] BREWER, J.; DESIGN J. Extreme Programming FAQ. [Interaktyvus] [Žiūrėta 2009 05 01]. <http://www.jera.com/techinfo/xpfaq.html>

[38] ALLEN, M. Extreme Programming FAQ. [Interaktyvus] [Žiūrėta 2009 05 01]. <Extreme Programming Summary Version 1.0.0>

[39] RICHARDSON, M. eXtreme Programming, Open Source, and Geographically Dispersed teams. [Interaktyvus] [Žiūrėta 2009 05 01]. <http://www.bsdcan.org/2004/papers/opensourcexp.pdf>

[40] TARABYKIN, A. Extreme Programming for Better Results on Software Development Projects. [Interaktyvus] [Žiūrėta 2009 05 01]. <http://www.pmforum.org/library/papers/2008/PDFs/Tarabykin-6-08.pdf>

[41] MAYFORD TECHNOLOGIES INC. Project Velocity. [Interaktyvus] [Žiūrėta 2009 05 01]. <http://www.mayford.ca/xp/velocity.html>

[42] THE STANDISH GROUP. Standish Chaos Reports. [Interaktyvus] [Žiūrėta 2009 05 09]. <http://www.standishgroup.com/newsroom/chaos_2009.php>

[43 ADDISON, T.; VALLABH, S. Controlling software project risks: an empirical study of methods used by experienced project managers. ISBN:1-58113-596-3 [Interaktyvus] 2002 134p. [Žiūrėta 2009 05 09] Prieiga per ACM Digital library

<http://portal.acm.org/citation.cfm?id=581506.581525&coll=Portal&dl=ACM&CFID=34892327&CFTOKEN=72027425>

[44] Ribinis programavimas [Žiūrėta 2009-05-16], prieiga per Internetą < http://lt.wikipedia.org/wiki/Ribinis_programavimas>

[45] Programinės įrangos kūrimo procesai [Žiūrėta 2009-05-16], prieiga per Internetą <http://vaidila.vdu.lt/~i5dasi/se2/paskaitos/02_procesai.pdf>

8. TERMINŲ IR SANTRUMPŲ ŽODYNAS

Ektremalus programavimas (Extreme Programming) – lanksčiojo programavimo metodika, dar vadinama ribiniu programavimu.

Kliento modulis – programa veikianti vartotojo kompiuteryje.

DBVS – duomenų bazių valdymo sistema.

CRM – santykių su klientais valdymo sistemos (Customer relationship management). Pvz.: OpenCRM, SugaCRM.

White-box – testavimo tikslas yra naudoti programos kodo žinias kuriant (projektuojant) testus.

Idealus laikas (Ideal time) – metrika nusakanti darbo laiką, be trukdymų, skirta darbo atlikimui.

Kalendorinis laikas (Calendar time) – maksimalus laikas, kuris gali būti skirtas darbui.

Užimtumo koeficientas (LoadFactor) – santykinis dydis tarp kalendorinio laiko ir idealaus laiko.

Projekto greitis (Project Velocity) – realizuotų vartotojo pasakojimų skaičius iteracijoje.

Vartotojo pasakojimas (User Story) – vartotojo išsakytas funkcinis reikalavimas vienu ar dviem sakiniais.

„**Vakarykštis oras**“ (**Yesterday's weather)** – metodas, kuriuo remiantis nusakoma kiek truks sekanti iteracija remiantis anksčiau darytos iteracijos duomenimis

Vartotojo pasakojimo kortelė (User Story Card) – formalios istorijų kortelės naudojamos aprašyti vartotojo pasakojimo istoriją.

„Paauksavimas“ (Gold plating) – reiškinys apibrėžiamas, kaip technologinio „tobulumo“ siekimas realizuojant funkcijas nenumatytas specifikacijoje. Pasireiškia realizavimo metu, kai nėra tiksliai apibrėžti reikalavimai.

Javascript – objektiškai orientuota skriptų programavimo kalba, besiremianti prototipų principu.

Agile programming – lankstusis programavimas.

Outsourcing – procesas, kurio metu viena organizacija samdo kitą organizaciją atlikti užduotis, kaip alternatyvą vidinių žmogiškųjų išteklių panaudojimui.

RSS – XML failų formatų šeima internetiniam duomenų rinkimui iš naujienų portalų ir tinklaraščių.

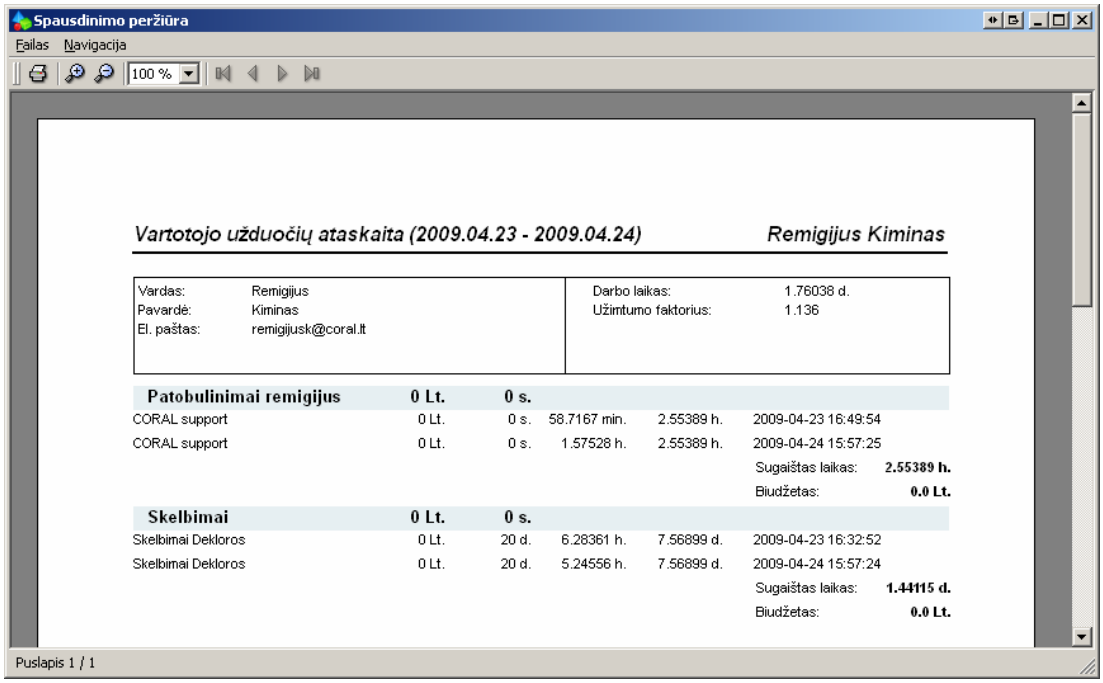
Zend Framework – PHP programavimo karkasas.

PHP – plačiai paplitusi dinaminė interpretuojama programavimo kalba.

9. PRIEDAI

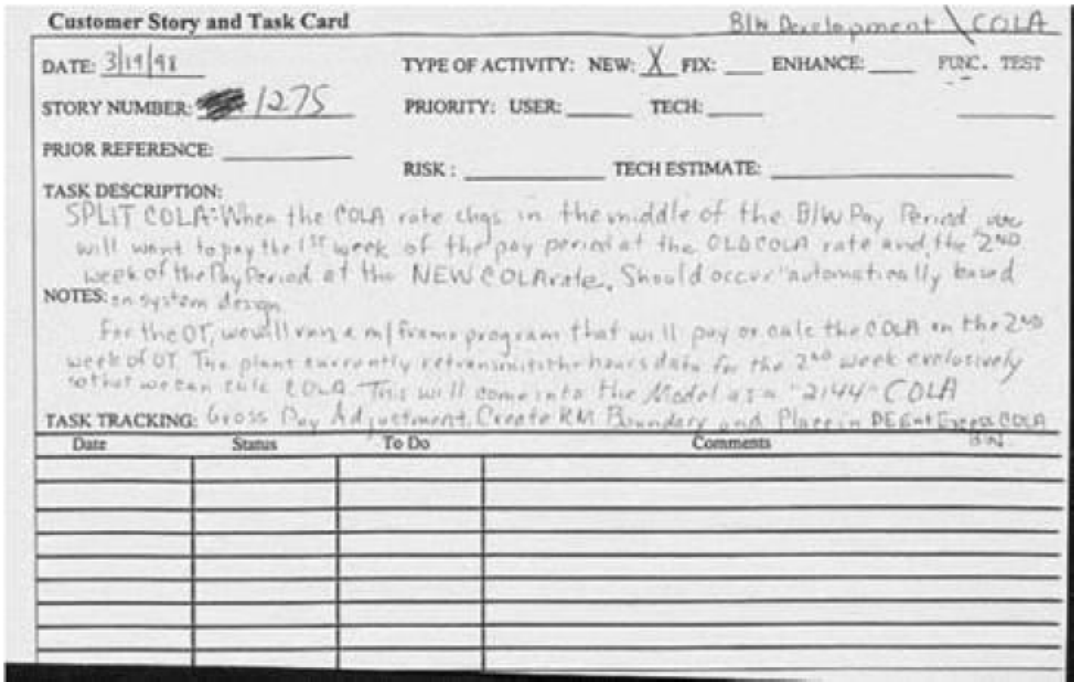
9.1. Idealaus laiko ataskaitos pavyzdys

Vartotojo idealaus laiko skaičiavimo ataskaita, kurioje pateikiamas automatiškai apskaičiuotas užimtumo koeficientas.



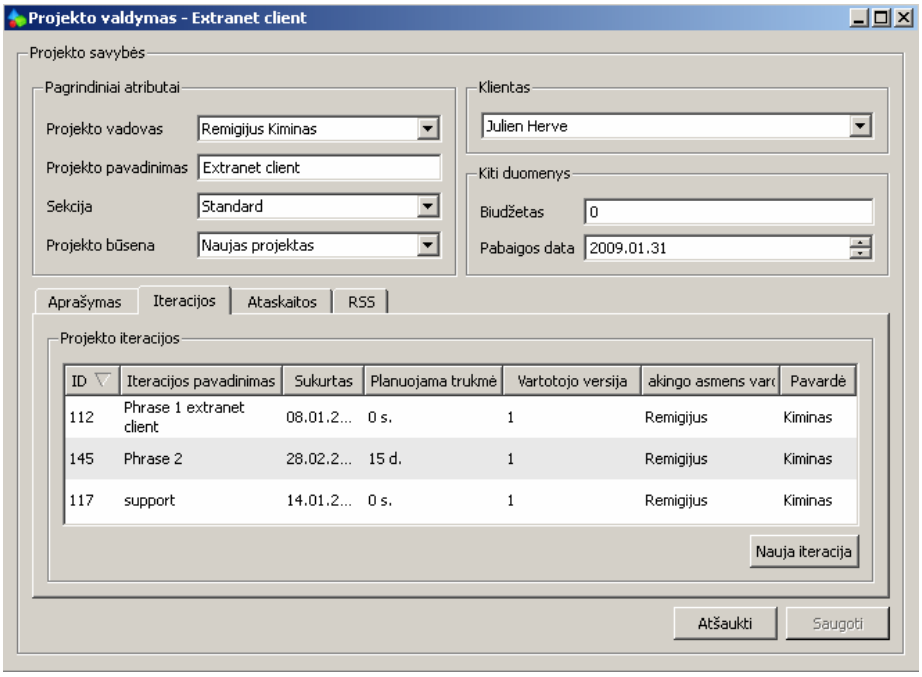
18 pav. Žurnalo funkcijos realizacija

9.2. Istorijos kortelės pavyzdys



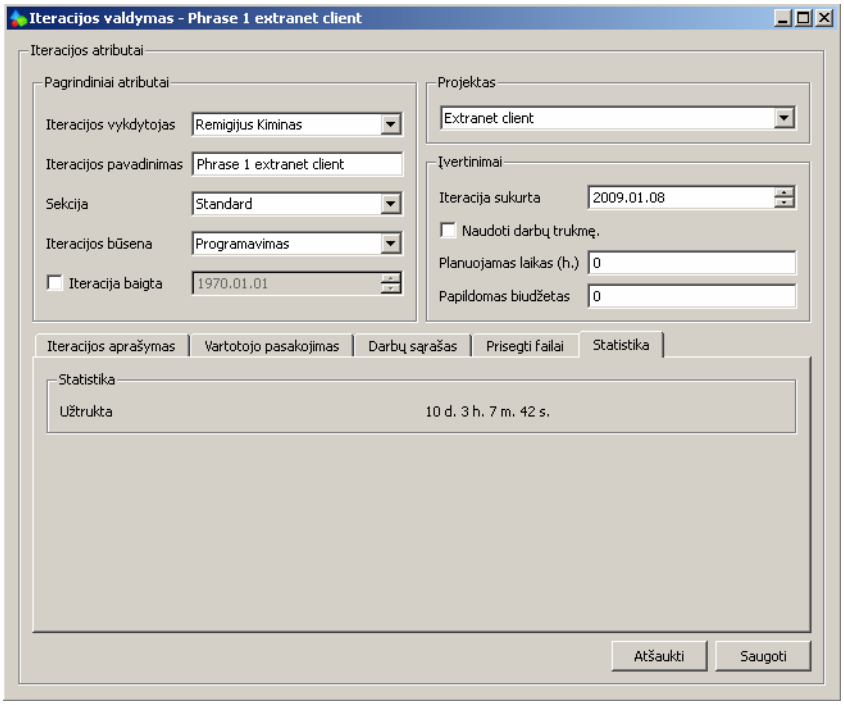
19 pav. Istorijos kortelės pavyzdys [34]

9.3. Projekto langas



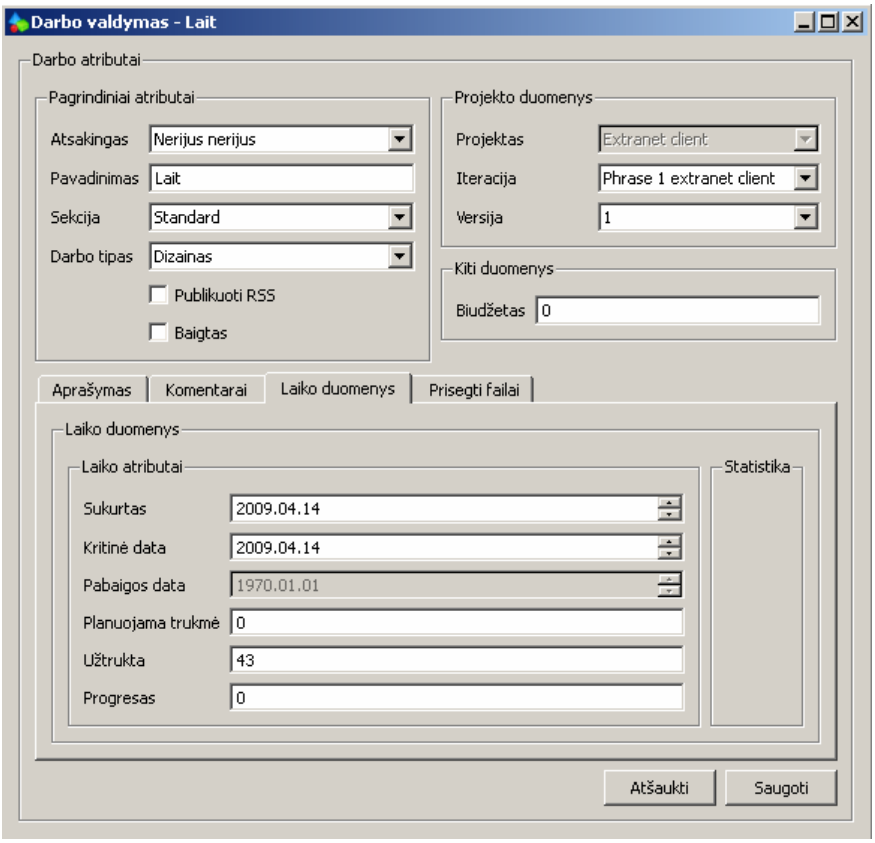
20 pav. Projekto lango ekrano nuotrauka

9.4. Iteracijos langas



21 pav. Iteracijos lango ekrano nuotrauka

9.5. Darbo langas



22 pav. Darbo lango ekrano nuotrauka

9.6. Programinės įrangos diegimo aktas

Pateikiamas programinės įrangos diegimo aktas.

9.7. Mokslinė veikla

Dalyvauta:

14-ji magistrantų ir doktorantų konferencija INFORMACINĖS TECHNOLOGIJOS.

Pristatyto straipsnio pavadinimas:

CSS elementų hierarchinių priklausomybių vaizdavimo modelis.

CSS ELEMENTŲ HIERARCHINIŲ PRIKLAUSOMYBIŲ VAIZDAVIMO MODELIS

Remigijus Kiminas[1],

[1]Kauno technologijos universitetas, informatikos fakultetas, Studentų g. 50, Kaunas, Lietuva, remdex@gmail.com

Abstraktas. Šis dokumentas aprašo sukurtą CSS elementų hierarchinių priklausomybių modelį. Detalizuojamas sukurtas prototipinis įrankis analizuojantis stilių tarpusavio priklausomybes ir pateikiantis jas grafiniu pavidalu. Buvo sudarytas pagrindinių stilių selektorių modelis skirtas grafiniam vaizdavimui. Sukurto prototipinio įrankio tikslas – padėti žiniatinklio programuotojams lengviau perprasti naudojamų stilių tarpusavio priklausomybes.

Raktažodžiai: CSS, Cascading Style Sheets, Graphviz, CSSTidy, hierarchija

10. Įžanga

Šių dienų puslapių didžioji išvaizdos dalis suformuoja naudojantis CSS taisyklėmis [8]. Dirbant su dideliais puslapiais iškylą problemų analizuojant stilius. Dažniausiai stilių failas užima kelis šimtus eilučių, o kartais ir kelis tūkstančių eilučių. Iškyla problemų su stilių analize. Yra sukurta nemažai įrankių skirtų HTML ir XML dokumentų redagavimui, tačiau yra sukurta sąlyginai mažai įrankių akcentuotų į stilių redagavimą [7]. Šiuo metu egzistuoja keli įrankiai, perteikiantys stilių tarpusavio savybes tam tikro elemento atžvilgiu [1]. Vienas iš populiariausių tokių įrankių yra „Firefox“ naršyklės „Firebug“ įskiepis. Jis leidžia pažiūrėti taikomus stilius tekstiniu pavidalu konkretaus elemento atžvilgiu. Tačiau nėra įrankių, leidžiančių vizualiai perteikiančių stilių tarpusavio priklausomybes [6]**.** Jei egzistuotų įrankiai, leidžiantys perteikti stilių priklausomybes, būtų galima daug greičiau perprasti paveldėtų sistemų išvaizdos formavimo savybes, aprašytų stilių priklausomybes. Šiam puslapių kūrimo aspektui buvo sukurtas prototipinis įrankis, leidžiantis vizualiai perteikti puslapio stilių tarpusavio priklausomybes ir įvertinti galimus jo panaudojimo aspektus.

11. CSS Elementų vaizdavimo modelis

CSS 2.1 aprašo virš 15 [2] elementų pasirinkimo taisyklių. Maišant stilių taikymo galimybes, įvairiems hierarchijos lygiams, gaunamos tam tikros puslapio išvaizdos valdymo savybės. Įmanoma aprašyti viduje esančius elementus, nekeičiant HTML kodo ir nepriskiriant atitinkamų klasių kiekvienam elementui pasinaudojant (Descendant selectors) savybėmis. Ši savybė dažniausiai naudojama su klasių parinktimis. Reikia pastebėti, kad dažniausiai iškyla problemų su stilių rašymu, esant jau aprašytiems stiliams, atėjusiems iš paveldėtų sistemų.

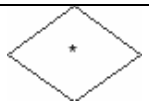
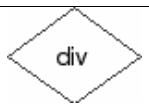
11.1. CSS Elementų žymėjimas

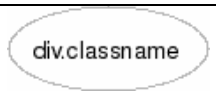
Prieš pradedant kurti prototipą pirmiausia turėjo būti sugalvotas stilių vaizdavimo modelis. Buvo sudarytas pagrindinių CSS elementų selektorių vaizdavimo modelis. Sukurtas prototipas realizavo keturis CSS elementų stilių aprašymo būdus [2].

2. * – stilius aprašomas visiem elementams
3. E – stilius aprašomas šio tipo elementams
4. E.classname – stilius aprašomas E tipo elementui su priskirta klase
5. E#classid – stilius aprašomas E tipo elementui su priskirtu identifikatoriumi

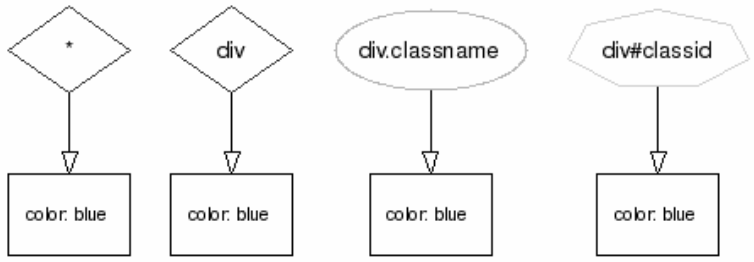
Buvo sudarytas kiekvieno iš jų vaizdavimo modelis:

Lentelė 1. CSS Elementų grafinis žymėjimas

Tipas	Žymėjimas
*	
E	

Tipas	Žymėjimas
E.classname	
E#classid	

Pavyzdys vaizduojantis visus keturis elementų tipus:



Paveikslėlis 1. Elementų vaizdavimo pavyzdys

Kuriant elementų žymėjimo notaciją nebuvo rastą analogų grafiniam CSS taisyklių vaizdavimo modeliui. Apžvelgti CSS taisyklių redagavimo įrankiai akcentuojami į tekstinę stilių pateikimo, redagavimo formą Pvz. „Style Master“, „Rapid CSS 2008“ ir kiti. Sudarant modelio notaciją buvo siekiama kuo suprantamesne forma pateikti vartotojui stilių tarpusavio priklausomybes.

11.2. CSS Elementų ryšiai

Taikant stilius įvairiems puslapio elementams susiduriama su stilių paveldėjimais. Aprašant stilius ,pasinaudojus stilių paveldėjimais, taupomas puslapio dydis, kadangi nėra būtina nurodyti klasių pavadinimus kiekvienam html kodo elementui ir dėlto mažėja HTML kodo dydis. Tokiu būdu aprašyta puslapio išvaizda leidžia sutaupyti HTML kodo stilių paprastumo sąskaita. Pasinaudojus šia CSS savybe užtenka aprašyti stilius aukštesniam lygmeniui einant hierarchijos tvarka žemyn. Žemiau pateiktame pavyzdyje (1) išryškėja bene dažniausiai sutinkama problema. Antrasis aprašytas stilius neturi jokios įtakos ir naudojamas aukštesniam hierarchijos lygyje aprašytas stilius. Nuorodos spalva bus raudonos spalvos. Tačiau jei nėra aprašytų stilių aukštesniame lygyje, taikomi žemesniame lygyje aprašyti stiliai, todėl nuoroda bus paryškinta. Šis aspektas ypač išryškėja dirbant su dideliais CSS failais, kur stiliai būna aprašyti keliose vietose. Norint išsiaiškinti stilių paveldėjimus, reikia naudoti „Firebug“ įskiepį, tačiau jį naudoti kiekvieną kartą nepatogu.

Pavyzdys 1.

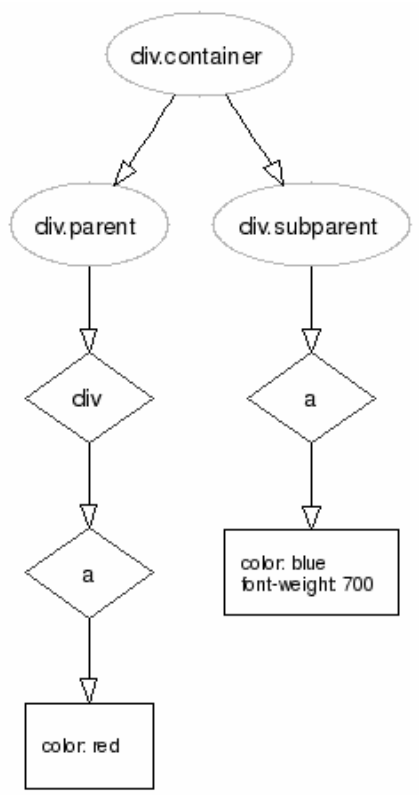
Stilių pavyzdys:

```
div.container div.parent div a
{
        color:red;
}
div.container div.subparent a
{
        color: blue;
        font-weight:bold;
}
```

HTML kodo pavyzdys:

```
<div class="container">
   <div class="parent">>
      <div class="subparent">
        <a href="# ">Nuoroda</a>
      </div>
   </div>
</div>
```

Nereikia pamiršti, kad aprašyti stiliai aukštesniam lygmeniui automatiškai taikomi ir žemesniam lygmeniui. Dirbant su paveldėtomis sistemomis, labai dažnai tenka susidurti su būtent tokiu būdu aprašytais stiliais ir tenką nemažą laiko dalį skirti stilių analizei, kadangi keičiant stilius, jie nesikeičia. Šiai problemai spręsti ir buvo sukurtas prototipas, kuris galėtų palengvinti šią analizę. Prototipo pagalba puslapio hierarchijos grafikas būtų sugeneruotas vieną kartą ir naudojamas tolimesniuose kūrimo etapuose. Aprašyto pavyzdžio prototipo sugeneruotas grafikas:



Paveikslėlis 2. Sugeneruotas grafikas

Įrankio naudingumas išryškėja dirbant su daug didesniais stiliais, perprantant puslapio stilių hierarchijos koncepciją. Grafike aukštesnio lygmens atributai būna grupuojami ir jo pagalba galima lengvai atsekti taikomus stilius.

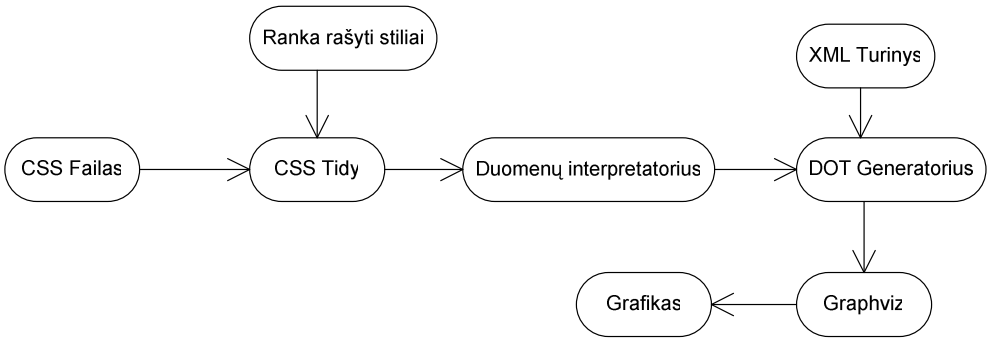
11.3. Prototipo realizacija

Prototipo realizacijai buvo pasirinktas jau sukurtas CSS stilių interpretatorius CSS Tidy [3]. Šis įrankis paskutinį kartą buvo atnaujintas tik 2007 metais, todėl ne visos CSS taisyklės korektiškai interpretuojamos. Įrankio tiesioginė paskirtis – aprašytų stilių optimizavimas, bet kaip parodė sukurtas prototipas, įrankis gali būti panaudotas ir platesniems tikslams.

Grafiniai vaizdavimui buvo panaudotas „Graphviz“ [4] įrankis. Pastarasis įrankis yra naudojamas daugelyje programinės įrangos produktų ir turi keletą sąsajų su įvairiomis programavimo kalbomis. Įrankis pasižymi lanksčia grafikų generavimo sintakse, kuri leidžia realizuoti įvairių formų grafikus, kas ir buvo panaudota kuriant prototipą. Grafikai aprašomi specialia DOT sintakse [5]. Realizuotas įrankis sugeba generuoti grafikus iš dviejų tipų duomenų:

1. CSS Taisyklių
2. XML Failų

Sukurto prototipo principinė schema:



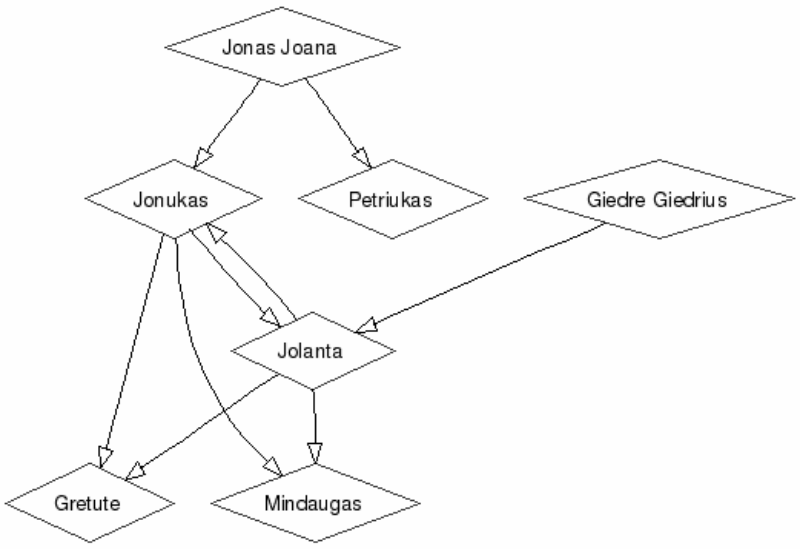
Paveikslėlis 3. Prototipo veikimo schema

Sukurtam prototipui duomenys gali būti pateikiami iš CSS failo, ranka rašytų stilių ar XML dokumento. Pateikus duomenis iš CSS failo ar tiesiai įvestų CSS taisyklių, būtinas tarpinis duomenų interpretavimas. DOT generatorius sugeneruoja sintaksę Graphviz įrankiui. Graphviz įrankio pasirinkimą lėmė jo platus taikymas dokumentacijos generavimo įrankiuose. Jis naudojamas grafikų braižymui Doxygen, phpDocumentor PĮ paketuose. Kadangi Graphviz plačiai naudojamas dokumentacijos generavimo paketuose buvo nuspręstą, kad jis geriausiai atitiks kuriamo prototipo funkcinius reikalavimus.

12. Hierarchijos vaizdavimo ypatumai

Apžvelgiant sukurto įrankio vaizdavimo savybes, reikia pastebėti, kad dabartinis įrankis korektiškai atvaizduoja keturias anksčiau paminėtas stilių priklausomybes. Šiuo metu generuojant grafikus nėra atsižvelgiama į HTML kodą, dėl šio trūkumo ne visada galima atsekti konkretaus elemento naudojamus stilius. DOT generatorius turėtų būti praplėstas HTML kodo interpretatoriumi, kuris nustatytų ryšį tarp stilių taisyklių ir HTML kodo.

Kadangi DOT generatorius palaiko ir XML failus, jo savybes galima lengvai panaudoti ir kitose srityse. Buvo sudarytas bandomasis šeimos medžio grafikas nekeičiant DOT generatoriaus kodo. Gautas šis pavyzdinis šeimos grafikas iš parašyto XML kodo:



Paveikslėlis 4. Šeimos medis

13. Išvados

Sukurtas prototipas leidžia lengviau perprasti anksčiau aprašytų stilių failus. Generuojami esminiai puslapio aprašytų stilių grafikai. Grafikų generavimas galimas ne vien iš CSS aprašytų stilių, bet ir iš XML. Generavimo algoritmą galima pritaikyti ir kitoms sritims, kuriose dominuoja hierarchinės priklausomybės. Tolimesnis įrankio tobulinimas įgalintų platesnę CSS selektorių tipų analizę ir jų priklausomybę nuo HTML kodo. Dabar analizuojami tik CSS stiliai jų pačių atžvilgiu. Visapusiškam įrankio realizavimui būtinad HTML kodo analizės susiejimas su aprašytais stiliais. Egzistuojantis Firefox naršyklės įskiepis „Firebug“ pateikia tokią priklausomybę, tačiau atvaizduoja ją tik konkretaus elemento atžvilgiu. Pilnai realizuotas įrankis leistų dar greičiau perprasti visą puslapio struktūrą. Pasinaudojus prototipu būtų galima praplėsti „Firebug“ įskiepio galimybes ir pateikti stilių hierarchijos grafikus ne vien tekstine, bet ir grafine forma.

Prototipą kartu su kodu galima rasti adresu http://css2uml.remdex.info.

Literatūra

[1] **Lerner M.** Linux Journal. *At the Forge: Firebug*, 2007. Issue 157, 8p., ISSN: 1075-3583

[2] **BOS B., CELIK .T3, WIUM LIE H. C** Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. *Selecors,* [Interaktyvus] 2007. [Žiūrėta 2009 03 11]. Prieiga internetu <http://www.w3.org/TR/CSS21/selector.html>

[3] **SCHMITZ F.** CSSTidy. [Žiūrėta 2009 03 11]. Prieiga internetu <http://csstidy.sourceforge.net>

[4] **AT&T Research.** Graphviz. [Žiūrėta 2009 03 11]. Prieiga internetu < http://www.graphviz.org >

[5] **GANSNER E., KOUTSOFIOS E., NORTH S.** Drawing graphs with dot. *2006.* [Žiūrėta 2009 03 11]. Prieiga internetu < http://www.graphviz.org/pdf/dotguide.pdf>

[6] **BADROS G., MARRIOTT K., BORNING A., STUCKEY P.** Constraint Cascading Style Sheets for the Web. *1999.* [Žiūrėta 2009 04 08]. Prieiga internetu <http://portal.acm.org/citation.cfm?id=320719.322588&coll=Portal&dl=ACM&CFID=29465031&CFTOKEN=97543819>

[7] **QUINT V., VATTON I.** Editing with Style. *2007.* [Žiūrėta 2009 04 08]. Prieiga internetu <http://portal.acm.org/citation.cfm?id=1284420.1284460&coll=Portal&dl=GUIDE&CFID=29467899&CFTOKEN=80359466>

[8] **JACOBS I., WALSH I.** Architecture of the World Wide Web, Volume One. *2004.* [Žiūrėta 2009 04 08]. Prieiga internetu <http://www.w3.org/TR/webarch>